

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH



NGÔ TRẦN QUỐC BẢO

BÀI ĐỒ XE THÔNG MINH ỨNG DỤNG IOT VÀ
TÍCH HỢP MACHINE LEARNING NHẬN DIỆN VỊ
TRÍ CHỖ TRÔNG TRONG BÀI ĐỒ XE

BÁO CÁO ĐỒ ÁN 1
NGÀNH HỆ THỐNG NHÚNG VÀ IOT

TP. Hồ Chí Minh, Tháng 05, 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH

NGÔ TRẦN QUỐC BẢO

BÀI ĐỒ XE THÔNG MINH ỨNG DỤNG IOT VÀ
TÍCH HỢP MACHINE LEARNING NHẬN DIỆN VỊ
TRÍ CHỖ TRỐNG TRONG BÀI ĐỒ XE

NGÀNH HỆ THỐNG NHÚNG VÀ IOT

Người hướng dẫn khoa học: TS. Huỳnh Thế Thiện

TP. Hồ Chí Minh, Tháng 05, 2025

Lời Cam Đoan

Tôi, Ngô Trần Quốc Bảo, sinh viên ngành Hệ thống nhúng và IoT, mã số sinh viên 22139004, khoa Điện - Điện tử, trường Đại học Sư phạm Kỹ thuật thành phố Hồ Chí Minh, xin cam đoan rằng đồ án 1 với đề tài này là sản phẩm nghiên cứu độc lập và sáng tạo của bản thân. Đồ án được hoàn thành dựa trên nền tảng kiến thức chuyên môn, kỹ năng nghiên cứu và nỗ lực cá nhân của tôi, không sao chép nội dung hay kết quả của bất kỳ đồ án, bài báo khoa học, hay tài liệu nào khác đã được công bố trước đây. Tất cả nội dung tham khảo từ các nguồn tài liệu khác, bao gồm sách, báo, bài báo khoa học, trang web, v.v., đều được trích dẫn đầy đủ và chính xác theo đúng quy định của trường và các tiêu chuẩn học thuật quốc tế. Tôi cam đoan rằng thông tin trong đồ án là trung thực và khách quan, không có hành vi gian lận hay đạo văn dưới bất kỳ hình thức nào. Tôi nhận thức được hậu quả của việc vi phạm cam đoan này và sẵn sàng chịu trách nhiệm trước hội đồng và nhà trường nếu có bất kỳ sai phạm nào được phát hiện.

Người thực hiện tiểu luận
(Ký và ghi rõ họ tên)

Ngô Trần Quốc Bảo

Lời Cảm Tạ

Trong quá trình hoàn thành đồ án 1 với đề tài này, tôi đã nhận được rất nhiều sự tư vấn và lời khuyên từ quý Thầy Cô và bạn bè ngành Hệ thống nhúng và IoT, khoa Điện - Điện tử, thuộc trường Đại Học Sư Phạm Kỹ Thuật Thành phố Hồ Chí Minh mới có thể hoàn thành đề tài. Tôi xin gửi một lời cảm ơn chân thành nhất đến những quý Thầy Cô, bạn bè đã góp ý, giúp đỡ trong quá trình thực hiện đề tài. Đặc biệt, tôi xin cảm ơn Thầy Huỳnh Thế Thiện, người đã luôn sát cánh cùng tôi trong quá trình thực hiện đề tài. Những góp ý, lời khuyên, đánh giá và lộ trình Thầy đặt ra từ những ngày đầu làm đồ án thực sự để lại ý nghĩa rất lớn, ảnh hưởng đến tư duy, hướng phát triển trong quá trình thực hiện đề tài này của tôi. Tất nhiên, đề tài sẽ còn có những thiếu sót và hạn chế, tôi rất mong nhận được những góp ý và đánh giá thẳng thắn để có thể học hỏi thêm kiến thức, qua đó có thể có một sản phẩm chỉn chu hơn trong tương lai. Tôi xin chân thành cảm ơn!

Tóm Tắt

Đề tài này hướng đến việc xây dựng một hệ thống hỗ trợ quản lý và giám sát bãi xe hiệu quả, hiện đại. Hệ thống sử dụng vi điều khiển ESP8266 kết hợp với module đọc thẻ RFID RC522 để nhận diện và kiểm soát phương tiện ra vào. Bên cạnh đó, đề tài tích hợp camera và ứng dụng kỹ thuật xử lý ảnh cùng mô hình Machine Learning nhằm xác định và phân loại các vị trí trong bãi xe như: vị trí đã đỗ, còn trống và đỗ sai quy định. Dữ liệu từ hệ thống được lưu trữ và cập nhật thời gian thực thông qua Firebase Realtime Database, giúp việc theo dõi và thống kê trở nên dễ dàng và linh hoạt. Để nâng cao khả năng tương tác và trực quan hóa, một dashboard quản lý được phát triển bằng Django, cho phép người dùng theo dõi tình trạng bãi xe, hình ảnh camera, lịch sử xe ra vào, và thông tin chi tiết được đồng bộ tự động.

Mục Lục

1 TỔNG QUAN	1
1.1 GIỚI THIỆU ĐỀ TÀI	1
1.2 MỤC TIÊU NGHIÊN CỨU CỦA ĐỀ TÀI	2
1.3 PHƯƠNG PHÁP NGHIÊN CỨU	2
1.4 PHẠM VI VÀ GIỚI HẠN CỦA ĐỀ TÀI	3
1.5 BỐ CỤC BÁO CÁO	5
2 CƠ SỞ LÝ THUYẾT	6
2.1 GIỚI THIỆU VỀ HỆ THỐNG IOT VÀ VI ĐIỀU KHIỂN ESP8266	6
2.1.1 Tổng quan về hệ thống IoT - Mạng lưới kết nối Internet vạn vật	6
2.1.2 Vi điều khiển ESP8266	7
2.2 CÔNG NGHỆ NHẬN DẠNG BẰNG THẺ RFID	9
2.3 Cơ sở về Firebase Realtime Database	11
2.3.1 Đặc điểm nổi bật	11
2.3.2 Kiến trúc và cách hoạt động	12
2.3.3 Ứng dụng trong đề tài	12
2.3.4 Hạn chế	13
2.4 TỔNG QUAN VỀ THỊ GIÁC MÁY TÍNH, CAMERA GIÁM SÁT VÀ XỬ LÝ ẢNH CƠ BẢN	13
2.5 PHÂN LOẠI ẢNH VỚI SUPPORT VECTOR CLASSIFIER (SVC)	14

2.6	CƠ SỞ XỬ LÝ ẢNH BẰNG MASK VÀ PHÂN VÙNG ĐỊNH TRƯỚC TRONG PHÂN TÍCH BÃI ĐỖ XE	16
2.7	TỔNG QUAN VỀ JDANGO VÀ XÂY DỰNG GIAO DIỆN DASHBOARD	17
3	THIẾT KẾ HỆ THỐNG	18
3.1	YÊU CẦU CỦA HỆ THỐNG	18
3.2	TỔNG QUAN KIẾN TRÚC HỆ THỐNG	19
3.3	THIẾT KẾ PHẦN CỨNG: ESP8266, MODULE RFID RC522 VÀ TRUYỀN DỮ LIỆU .	20
3.4	THIẾT KẾ PHẦN MỀM: GIAO DIỆN DASHBOARD SỬ DỤNG DJANGO, LUU TRỮ VÀ QUẢN LÍ DỮ LIỆU BẰNG FIREBASE	23
3.5	HỆ THỐNG PHÂN TÍCH TRẠNG THÁI BÃI ĐỖ XE	25
3.5.1	Phát hiện tình trạng chỗ trống bằng mặt nạ định trước (mask)	25
3.5.2	Mô phỏng video thay thế camera thực tế	26
4	TRIỂN KHAI HỆ THỐNG THỰC TẾ	28
4.1	QUY TRÌNH TRIỂN KHAI HỆ THỐNG	28
4.2	HUẤN LUYỆN VÀ ĐÁNH GIÁ MÔ HÌNH HỌC MÁY	29
4.2.1	Mô tả dữ liệu đầu vào và quá trình huấn luyện SVC	29
4.2.2	Triển khai mô hình nhận diện vị trí đỗ xe	31
4.2.3	Dánh giá độ chính xác mô hình nhận diện vị trí đỗ xe	32
4.3	TRIỂN KHAI KẾT NỐI THIẾT BỊ THỰC TẾ VÀ HỆ THỐNG DASHBOARD	32
5	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	35
5.1	KẾT LUẬN	35
5.2	ĐỀ XUẤT CẢI TIẾN VÀ MỞ RỘNG HỆ THỐNG TRONG TƯƠNG LAI	36

Danh sách hình

3.2.1 Mô hình hoạt động của hệ thống	20
3.3.2 Sơ đồ kết nối ESP8266 và Module RFID RC522	21
3.5.3 Mặt nạ định trước (mask) bao gồm các phân vùng tương ứng với từng ô đỗ xe trong bãi .	26
3.5.4 Minh họa khung hình trích từ video mô phỏng bãi đỗ xe	26
4.2.1 Bản đồ vị trí chỗ đỗ xe (Parking Map) hiển thị trạng thái Empty và Not Empty	33
4.3.2 Minh họa Dashboard bãi đỗ xe thông minh ứng dụng iot và tích hợp machine learning nhận diện vị trí chỗ trống trong bãi đỗ xe	34

Danh sách bảng

2.1.1 Bảng mô tả các chân của vi điều khiển ESP8266 NodeMCU	8
3.3.1 Sơ đồ kết nối giữa RC522 và ESP8266	21

Danh sách các từ viết tắt

Các từ viết tắt	Định nghĩa
API	Application Programming Interface
IoT	Internet of Things
RFID	Radio Frequency Identification
SVC	Support Vector Classifier
UID	Unique Identifier

Chương 1

TỔNG QUAN

1.1 GIỚI THIỆU ĐỀ TÀI

Trong bối cảnh đô thị hóa ngày càng gia tăng, nhu cầu quản lý bãi đỗ xe một cách thông minh và hiệu quả trở thành một vấn đề cấp thiết. Các hệ thống truyền thống thường dựa vào con người để giám sát và ghi nhận thông tin xe vào – ra, dẫn đến nhiều bất cập như tốn công sức, dễ xảy ra sai sót, và không kịp thời trong việc phát hiện các tình huống đỗ xe sai quy định hoặc quá thời gian.

Đề tài "Bãi đỗ xe thông minh ứng dụng IoT và tích hợp Machine Learning nhận diện vị trí chỗ trống trong bãi đỗ xe" nhằm mục tiêu giải quyết bài toán trên bằng cách tích hợp các công nghệ hiện đại bao gồm: hệ thống nhận dạng bằng thẻ RFID kết hợp với vi điều khiển ESP8266 để thu thập dữ liệu xe vào – ra, lưu trữ và đồng bộ dữ liệu theo thời gian thực lên Firebase, và đặc biệt là ứng dụng thị giác máy tính để giám sát vị trí đỗ xe thông qua video mô phỏng. Hệ thống còn hỗ trợ giao diện quản lý trực quan thông qua dashboard web được xây dựng bằng Django, giúp người quản lý dễ dàng theo dõi trạng thái bãi xe.

Điểm nổi bật của đề tài là sử dụng mô hình học máy Support Vector Classifier (SVC) thay vì các mô hình học sâu phức tạp, nhằm đảm bảo tính đơn giản, dễ triển khai trong môi trường tài nguyên hạn chế nhưng vẫn đạt hiệu quả nhận diện tương đối tốt. Bên cạnh đó, việc sử dụng video mô phỏng có vòng lặp thay thế cho camera thực tế giúp quá trình thử nghiệm và kiểm thử hệ thống trở nên linh hoạt và tiết kiệm chi phí hơn trong giai đoạn nghiên cứu.

Với sự kết hợp giữa phần cứng IoT, xử lý ảnh và nền tảng web, hệ thống hướng đến việc xây dựng một mô hình bãi giữ xe thông minh, có khả năng mở rộng và ứng dụng thực tế trong tương lai.

1.2 MỤC TIÊU NGHIÊN CỨU CỦA ĐỀ TÀI

Mục tiêu chính của đề tài là xây dựng một hệ thống bãi đỗ xe thông minh dựa trên các công nghệ IoT, học máy và thị giác máy tính, giúp tự động hóa quá trình ghi nhận và giám sát xe ra vào cũng như vị trí đỗ xe. Cụ thể, đề tài hướng đến các mục tiêu sau:

- Tích hợp hệ thống nhận dạng xe bằng thẻ RFID sử dụng module RC522 và vi điều khiển ESP8266 để tự động ghi nhận thông tin xe ra vào.
- Lưu trữ và đồng bộ dữ liệu theo thời gian thực thông qua nền tảng Firebase Realtime Database, bao gồm UID thẻ RFID, biển số xe, thời gian và thông tin thanh toán.
- Xây dựng hệ thống nhận diện vị trí đỗ xe bằng cách ứng dụng xử lý ảnh từ video mô phỏng, kết hợp với mô hình học máy Support Vector Classifier (SVC) để phân loại trạng thái chiếm chỗ trong từng ô đỗ.
- Thiết kế giao diện quản lý web bằng Django nhằm hiển thị trực quan danh sách xe đang gửi, hình ảnh giám sát từ video, và thông tin chi tiết từng xe.
- Kết nối phần cứng với hệ thống phần mềm một cách liền mạch, đảm bảo dữ liệu từ thiết bị RFID được phản ánh tức thời lên giao diện dashboard.
- Dánh giá độ chính xác của mô hình phân loại vị trí đỗ xe, từ đó đề xuất hướng cải tiến hệ thống để nâng cao hiệu suất và khả năng triển khai thực tế.

Thông qua các mục tiêu trên, đề tài kỳ vọng mang đến một giải pháp khả thi, tiết kiệm chi phí và dễ triển khai trong việc quản lý bãi đỗ xe tại các khu dân cư, trường học, công ty hoặc các cơ sở có quy mô vừa và nhỏ.

1.3 PHƯƠNG PHÁP NGHIÊN CỨU

Để đạt được mục tiêu đã đề ra trong đề tài “Bãi đỗ xe thông minh ứng dụng IoT và tích hợp Machine Learning nhận diện vị trí chỗ trống trong bãi đỗ xe”, tác giả đã áp dụng các phương pháp nghiên cứu như sau:

- **Phương pháp nghiên cứu tài liệu:** Tìm hiểu, thu thập và tổng hợp kiến thức từ các nguồn tài liệu chính thống, sách chuyên ngành, các bài báo khoa học, tài liệu hướng dẫn kỹ thuật liên quan đến ESP8266, RFID, Firebase, xử lý ảnh và mô hình học máy SVC. Ngoài ra, tham khảo các hệ thống bãi giữ xe thông minh đã được triển khai thực tế để rút ra các điểm mạnh, hạn chế và khả năng ứng dụng.

- **Phương pháp thực nghiệm:** Tiến hành xây dựng mô hình hệ thống phần cứng gồm vi điều khiển ESP8266 và module RFID RC522. Cấu hình các thành phần để nhận diện UID của thẻ RFID và gửi dữ liệu lên Firebase. Đồng thời, xây dựng mô hình mô phỏng video camera nhằm phục vụ cho việc nhận diện và phân tích vị trí đỗ xe bằng xử lý ảnh.
- **Phương pháp mô phỏng và đánh giá mô hình học máy:** Thu thập dữ liệu ảnh/video mô phỏng bãi xe, gán nhãn thủ công và huấn luyện mô hình Support Vector Classifier (SVC) để phân loại trạng thái ô đỗ (có xe hoặc không). Đánh giá hiệu quả mô hình thông qua các chỉ số như độ chính xác (Accuracy), độ nhạy và độ đặc hiệu.
- **Phương pháp thiết kế và xây dựng phần mềm:** Thiết kế giao diện Dashboard bằng Django để hiển thị trực quan thông tin xe gửi, hình ảnh từ camera (video mô phỏng), và các dữ liệu thời gian thực từ Firebase. Giao diện cho phép quản trị viên dễ dàng theo dõi, kiểm tra và đánh giá tình trạng bãi xe.
- **Phương pháp kiểm thử và đánh giá hệ thống:** Kiểm thử toàn bộ hệ thống thông qua các kịch bản thực tế, đánh giá tính ổn định, khả năng kết nối giữa phần cứng và phần mềm, độ chính xác mô hình, và khả năng cập nhật dữ liệu thời gian thực.

1.4 PHẠM VI VÀ GIỚI HẠN CỦA ĐỀ TÀI

Đề tài này tập trung vào việc thiết kế và triển khai hệ thống bãi giữ xe thông minh, sử dụng các công nghệ IoT kết hợp với các phương pháp xử lý ảnh, và học máy để tự động hóa quy trình quản lý bãi đỗ xe. Hệ thống sử dụng ESP8266 để điều khiển mạch đọc thẻ RFID nhằm nhận diện và kiểm soát xe vào ra. Mô hình học máy SVC được áp dụng để phân loại các vị trí đỗ xe là trống hay đã đỗ, đồng thời phát hiện vị trí đỗ sai quy định dựa trên dữ liệu từ video thay vì sử dụng camera thực tế. Tuy nhiên trong khuôn khổ của nghiên cứu và thực nghiệm, đề tài vẫn còn một số giới hạn và phạm vi được đặt ra như sau:

1. **Hệ thống nhận diện không sử dụng camera thực tế:** Trong quá trình kiểm tra và nhận diện, hệ thống sử dụng hình ảnh tĩnh thay vì camera nhận diện khuôn mặt và biển số xe trong thực tế. Việc này giúp giảm chi phí triển khai, nhưng cũng làm giảm tính chính xác và khả năng ứng dụng trong môi trường thực tế với các tình huống thay đổi nhanh.
2. **Video thay vì camera trực tiếp:** Để huấn luyện mô hình học máy, video từ camera thực tế đã được sử dụng, nhưng khi hiển thị trên Dashboard, hệ thống chỉ sử dụng video đã được nhận diện các vị trí đỗ còn trống thay vì kết nối với camera trực tiếp. Điều này giúp đơn giản hóa quá trình kiểm thử và phát triển nhưng không phản ánh chính xác quy trình vận hành thực tế của hệ thống.
3. **Hạn chế về phần mềm và phần cứng:** Do giới hạn tài nguyên, phần cứng sử dụng trong đề tài chỉ có ESP8266 và các module RFID để thực hiện việc nhận diện xe ra vào, không bao gồm các hệ

thống nhận diện hình ảnh trực tiếp từ camera hoặc các cảm biến phức tạp khác.

4. **Mô hình học máy:** Mô hình học máy được sử dụng trong đề tài dựa trên phương pháp phân loại với SVC (Support Vector Classifier). Mặc dù mô hình này có thể phân loại chính xác các vị trí đỗ xe, nhưng do dữ liệu huấn luyện còn hạn chế, kết quả có thể không đạt được độ chính xác cao nhất trong các tình huống thực tế.
5. **Giới hạn về độ phân giải và kích thước ảnh:** Trong quá trình huấn luyện và phân loại, các ảnh được giảm xuống kích thước 15x15 pixel để giảm độ phức tạp của mô hình. Điều này có thể ảnh hưởng đến khả năng phân biệt các chi tiết nhỏ trong ảnh, đặc biệt là trong các trường hợp có sự thay đổi về ánh sáng hoặc góc độ.

Mặc dù có những hạn chế và giới hạn nhất định, hệ thống vẫn cung cấp một giải pháp khả thi cho việc giám sát và quản lý bãi đỗ xe thông minh trong môi trường thử nghiệm, đồng thời có thể mở rộng và cải tiến trong các nghiên cứu và ứng dụng thực tế sau này.

1.5 BỘ CỤC BÁO CÁO

Báo cáo đồ án 1 của tác giả sẽ bao gồm 5 chương:

- Chương 1: Tổng quan
- Chương 2: Cơ sở lý thuyết
- Chương 3: Thiết kế hệ thống
- Chương 4: Thực nghiệm và triển khai
- Chương 5: Kết luận và hướng phát triển.

Chương 2

CƠ SỞ LÝ THUYẾT

2.1 GIỚI THIỆU VỀ HỆ THỐNG IOT VÀ VI ĐIỀU KHIỂN ESP8266

2.1.1 Tổng quan về hệ thống IoT - Mạng lưới kết nối Internet vạn vật

Internet of Things (IoT) – Mạng lưới vạn vật kết nối Internet – là một khái niệm mô tả xu hướng công nghệ hiện đại, nơi các thiết bị vật lý được tích hợp cảm biến, phần mềm và kết nối mạng để thu thập, trao đổi dữ liệu với nhau thông qua Internet mà không cần sự can thiệp của con người. IoT mang lại khả năng kết nối và điều khiển các thiết bị từ xa một cách thông minh, qua đó giúp nâng cao hiệu quả, giảm chi phí vận hành và tối ưu hóa quy trình quản lý. Vì thế nên hiện nay hệ thống IoT đóng vai trò quan trọng trong việc tự động hóa và quản lý thông minh trong nhiều lĩnh vực như:

- **Nhà thông minh (Smart Home):** Điều khiển đèn, máy lạnh, khóa cửa qua smartphone.
- **Nông nghiệp thông minh:** Theo dõi độ ẩm đất, điều khiển tưới tiêu tự động.
- **Y tế:** Theo dõi nhịp tim, nhiệt độ cơ thể và gửi dữ liệu đến bác sĩ.
- **Giao thông:** Giám sát phương tiện, quản lý bãi đỗ xe, đèn giao thông thông minh.

Một hệ thống IoT thường bao gồm các thành phần chính sau:

- **Thiết bị cảm biến và thu thập dữ liệu:** Như cảm biến nhiệt độ, độ ẩm, RFID, camera,...
- **Thiết bị xử lý và truyền thông:** Thường là vi điều khiển hoặc máy tính nhúng, có khả năng xử lý tín hiệu và truyền dữ liệu.

- **Kết nối mạng:** Thông qua Wi-Fi, 3G/4G, Bluetooth hoặc ZigBee,...
- **Nền tảng lưu trữ và xử lý dữ liệu:** Thường sử dụng các dịch vụ đám mây như Firebase, AWS, ThingSpeak,...
- **Giao diện người dùng:** Giúp người dùng giám sát và điều khiển hệ thống (có thể là website, ứng dụng, dashboard,...)

Trong đề tài này, hệ thống IoT được ứng dụng để xây dựng **bãi đỗ xe thông minh**, trong đó các thiết bị như thẻ RFID, vi điều khiển, mô hình xử lý ảnh và hệ thống lưu trữ Firebase được kết nối và phối hợp để tự động hóa việc kiểm soát và giám sát phương tiện.

2.1.2 Vi điều khiển ESP8266

Trong hệ thống IoT, vi điều khiển đóng vai trò là trung tâm điều khiển và xử lý thông tin từ các cảm biến. Một trong những vi điều khiển phổ biến được sử dụng là ESP8266, nhờ vào tính năng tích hợp Wi-Fi, chi phí thấp, nhỏ gọn và khả năng lập trình linh hoạt [1].



Hình 2.1.1: ESP8266

ESP8266 là một vi điều khiển 32-bit được phát triển bởi hãng Espressif Systems (Trung Quốc). Với khả năng kết nối Internet và xử lý dữ liệu nhanh chóng, ESP8266 được ứng dụng rộng rãi trong các dự án IoT quy mô nhỏ đến trung bình [1]. Một số đặc điểm nổi bật của ESP8266 có thể kể đến như sau:

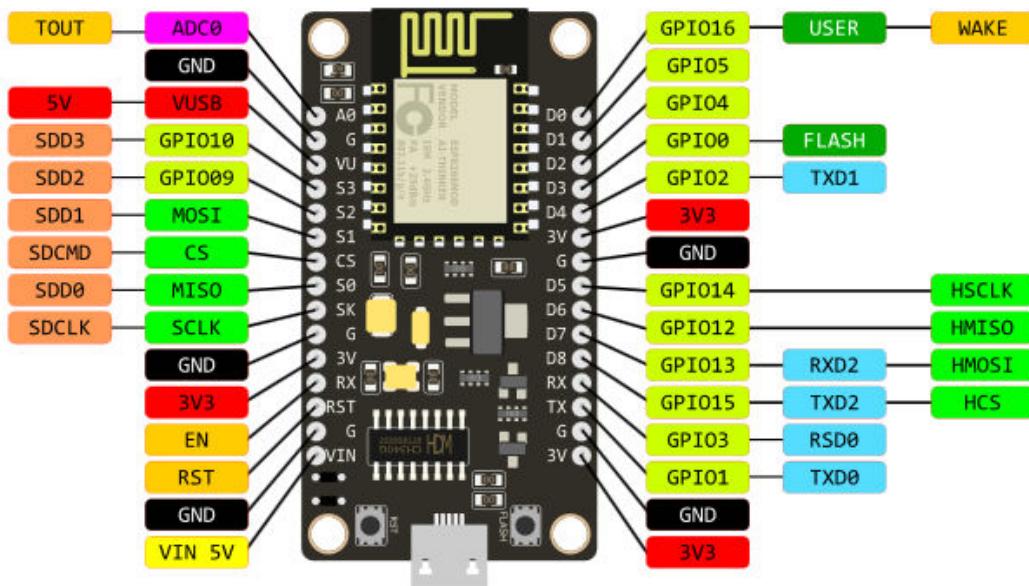
- Tích hợp Wi-Fi:** Hỗ trợ kết nối không dây chuẩn IEEE 802.11 b/g/n.
- Tốc độ xử lý:** Sử dụng CPU 32-bit Tensilica L106 với tốc độ lên đến 80 MHz.
- Bộ nhớ:** Có sẵn SRAM và flash tùy vào phiên bản (thường từ 512KB đến 4MB).
- Giao tiếp:** Hỗ trợ các chuẩn giao tiếp phổ biến như UART, SPI, I2C, PWM, GPIO,...
- Hỗ trợ lập trình:** Có thể lập trình bằng nhiều nền tảng như Arduino IDE, NodeMCU (Lua), MicroPython,...

Dưới đây là thông tin về các chân dùng để đấu nối khi kết hợp ESP8266 với các thiết bị phần cứng khác [1]

Bảng 2.1.1: Bảng mô tả các chân của vi điều khiển ESP8266 NodeMCU

Tên chân (Label)	GPIO	Chức năng chính	Ghi chú sử dụng
D0	GPIO16	Digital I/O, không hỗ trợ PWM/I2C	Không dùng cho interrupt
D1	GPIO5	Digital I/O, hỗ trợ I2C (SCL)	Nối với SCL nếu dùng I2C
D2	GPIO4	Digital I/O, hỗ trợ I2C (SDA)	Nối với SDA nếu dùng I2C
D3	GPIO0	Digital I/O, liên quan đến chế độ khởi động	Cần kéo lên mức cao khi boot
D4	GPIO2	Digital I/O, LED tích hợp (LED _{BUILTIN})	Cần kéo lên mức cao khi boot
D5	GPIO14	Digital I/O, hỗ trợ SPI (SCK)	Nối với SCK nếu dùng SPI
D6	GPIO12	Digital I/O, hỗ trợ SPI (MISO)	-
D7	GPIO13	Digital I/O, hỗ trợ SPI (MOSI)	-
D8	GPIO15	Digital I/O, liên quan đến boot	Cần kéo xuống mức thấp khi boot
RX	GPIO3	UART RX	Giao tiếp nối tiếp (Serial)
TX	GPIO1	UART TX	Giao tiếp nối tiếp (Serial)
3V3	-	Nguồn 3.3V cấp ra	Không cấp tải nặng
GND	-	Nối đất	Phải nối đất khi kết nối ngoại vi
VIN	-	Nguồn vào (từ 5V)	Có thể cấp từ USB hoặc pin ngoài

Trong đề tài này, ESP8266 (phiên bản NodeMCU) đóng vai trò là trung tâm điều khiển và truyền dữ liệu. Nó thực hiện các nhiệm vụ chính như: đọc và nhận dữ liệu từ mãđun RFID RC522 nhằm xác định xe vào/ra bãi, sau đó gửi dữ liệu UID đến hệ thống lưu trữ trực tuyến Firebase Realtime Database thông qua kết nối Wi-Fi, bên cạnh đó nó cũng sẽ giao tiếp với hệ thống giám sát để hiển thị thông tin về thời gian xe vào/ra bãi. Việc sử dụng ESP8266 giúp hệ thống hoạt động độc lập, giảm thiểu chi phí, đồng thời tăng khả năng triển khai thực tế nhờ vào tính linh hoạt và dễ lập trình và dễ tích hợp với các nền tảng phần mềm phía người dùng như Dashboard quản lý. Ngoài ra, nó còn phù hợp với các mô hình hệ thống nhỏ gọn và yêu cầu tiêu thụ năng lượng thấp.

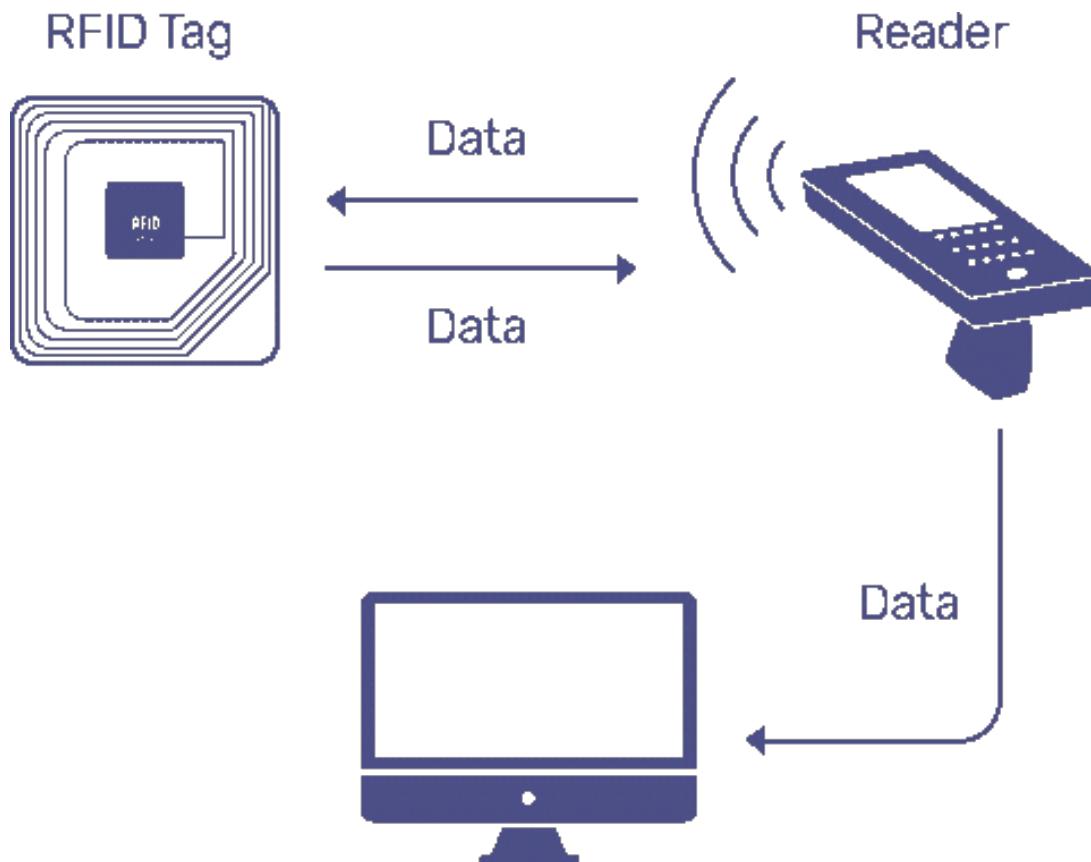


Hình 2.1.2: Sơ đồ đấu nối chân của ESP8266

2.2 CÔNG NGHỆ NHẬN DẠNG BẰNG THẺ RFID

RFID (Radio Frequency Identification – Nhận dạng qua tần số vô tuyến) là một công nghệ cho phép nhận dạng và thu thập dữ liệu từ các đối tượng bằng sóng vô tuyến mà không cần tiếp xúc[2]. Đây là công nghệ phổ biến trong các hệ thống kiểm soát ra vào, quản lý kho bãi, bãi đỗ xe thông minh, và nhiều ứng dụng IoT khác.

- **Thẻ RFID (Tag):** Gồm có chip nhớ lưu trữ dữ liệu (thường là một mã số định danh duy nhất - UID) và ăng-ten. Thẻ có thể là loại bị động (không cần nguồn điện riêng, sử dụng năng lượng từ đầu đọc) hoặc chủ động (có nguồn điện riêng).
- **Đầu đọc RFID (Reader):** Tạo ra tín hiệu radio để kích hoạt thẻ RFID và thu nhận dữ liệu phản hồi từ thẻ. Một số đầu đọc còn hỗ trợ giao tiếp với các thiết bị vi điều khiển hoặc máy tính để xử lý tiếp dữ liệu.
- **Hệ thống xử lý dữ liệu:** Thông qua kết nối với vi điều khiển, dữ liệu từ thẻ RFID sẽ được xử lý, lưu trữ hoặc gửi đến máy chủ, hệ thống quản lý trung tâm.



Hình 2.2.3: Trình bày nguyên lý hoạt động của module RFID

Nguyên lý hoạt động

1. Khi thẻ RFID tiến gần vùng quét của đầu đọc, đầu đọc phát ra tín hiệu tần số vô tuyến.
2. Thẻ RFID (loại bị động) nhận năng lượng từ tín hiệu này để kích hoạt chip.
3. Thẻ phản hồi lại dữ liệu (UID) thông qua sóng radio.
4. Đầu đọc thu tín hiệu và truyền dữ liệu này đến vi điều khiển (ví dụ: ESP8266) để xử lý hoặc lưu trữ.

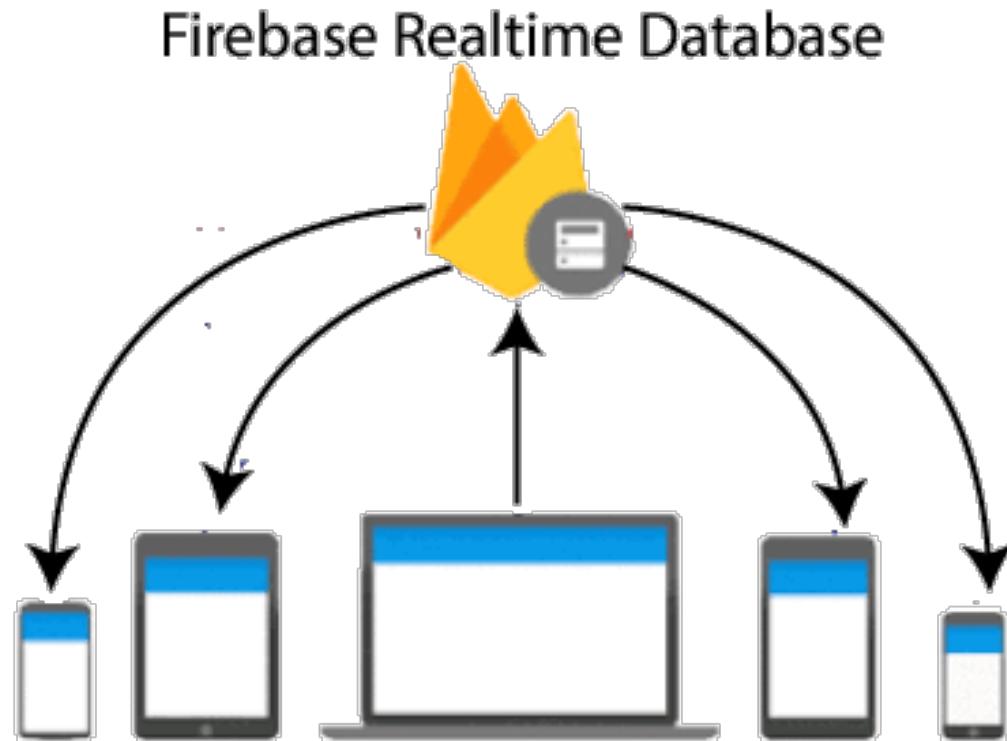
Ứng dụng trong đỗ tài

Trong hệ thống bãi đỗ xe thông minh, RFID được sử dụng để:

- **Nhận dạng phương tiện:** Mỗi thẻ RFID gắn với một người dùng hoặc phương tiện.
- **Tự động hóa quá trình check-in/check-out:** Khi thẻ được quét, hệ thống sẽ tự động ghi nhận thời gian ra vào và lưu thông tin lên cơ sở dữ liệu (Firebase).
- **Tăng độ chính xác và giảm thời gian xử lý:** Không cần thao tác thủ công, người dùng chỉ cần đưa thẻ lại gần đầu đọc để thực hiện thao tác.

Công nghệ RFID mang lại sự tiện lợi, chi phí thấp và tính mở rộng cao cho các hệ thống giám sát và quản lý phương tiện hiện đại.

2.3 Cơ sở về Firebase Realtime Database



Hình 2.3.4: Firebase Realtime Database

Firebase Realtime Database là một dịch vụ cơ sở dữ liệu NoSQL thời gian thực do Google phát triển, thuộc nền tảng Firebase. Cơ sở dữ liệu này lưu trữ dữ liệu ở dạng cây JSON, cho phép đồng bộ nhanh chóng giữa các client đang kết nối, phù hợp với các ứng dụng cần cập nhật liên tục và tức thời như hệ thống IoT[3].

2.3.1 Đặc điểm nổi bật

- **Cập nhật thời gian thực:** Bất kỳ thay đổi nào trong cơ sở dữ liệu đều được đồng bộ đến tất cả client đang kết nối mà không cần gửi lại yêu cầu.
- **Lưu trữ dưới dạng JSON:** Dữ liệu được tổ chức như một cây JSON, hỗ trợ truy cập nhanh qua các đường dẫn cụ thể.
- **Hỗ trợ đa nền tảng:** Firebase cung cấp SDK cho Android, iOS, Web và các nền tảng nhúng như ESP8266, giúp tích hợp dễ dàng.

- Bảo mật linh hoạt:** Hệ thống quy tắc bảo mật (Security Rules) cho phép định nghĩa chi tiết quyền truy cập theo người dùng hoặc điều kiện cụ thể.

2.3.2 Kiến trúc và cách hoạt động



Hình 2.3.5: Kiến trúc tổng quan của Firebase Realtime Database

Firebase hoạt động theo mô hình client-server, nơi mỗi thiết bị (như ESP8266) kết nối trực tiếp tới cơ sở dữ liệu thông qua WebSocket hoặc HTTP[3]:

- Thiết bị thực hiện kết nối tới đường dẫn có dạng <https://<tên-project>.firebaseio.com/>.
- Dữ liệu được gửi và nhận dưới định dạng JSON.
- Khi dữ liệu thay đổi tại bất kỳ nút nào trong cây JSON, Firebase sẽ chủ động cập nhật dữ liệu đó đến tất cả client đang lắng nghe tại vị trí đó.

2.3.3 Ứng dụng trong đề tài

Trong hệ thống bãi đỗ xe thông minh, Firebase Realtime Database hỗ trợ lưu trữ và hiển thị thông tin theo thời gian thực:

- Đồng bộ dữ liệu nhanh chóng:** Khi người dùng quét thẻ RFID, thông tin UID, thời gian, biển số,... được cập nhật ngay lập tức lên Firebase và hiển thị trên Dashboard.
- Tổ chức dữ liệu phân cấp:** Các thông tin như lịch sử xe vào/ra, trạng thái bãi xe,... được lưu trữ theo cấu trúc cây, thuận tiện truy xuất và thống kê.
- Triển khai đơn giản, tiết kiệm chi phí:** Firebase là dịch vụ đám mây nên không cần xây dựng hệ thống máy chủ riêng, phù hợp với các ứng dụng quy mô nhỏ đến trung bình.

2.3.4 Hạn chế

- **Hạn mức theo gói miễn phí:** Firebase giới hạn số lần đọc/ghi, băng thông và số kết nối đồng thời, có thể không đáp ứng nếu quy mô hệ thống lớn.
- **Khó thực hiện truy vấn phức tạp:** Do là cơ sở dữ liệu NoSQL, việc truy vấn theo điều kiện, lọc và sắp xếp nhiều trường trở nên hạn chế hơn so với cơ sở dữ liệu quan hệ.

Firebase Realtime Database là một giải pháp phù hợp cho các hệ thống cần cập nhật liên tục và phản hồi nhanh, đặc biệt trong các ứng dụng IoT như hệ thống bãi xe thông minh mà đề tài đang triển khai.

2.4 TỔNG QUAN VỀ THỊ GIÁC MÁY TÍNH, CAMERA GIÁM SÁT VÀ XỬ LÍ ẢNH CƠ BẢN

Thị giác máy tính (Computer Vision) là một lĩnh vực của trí tuệ nhân tạo (AI), chuyên nghiên cứu các phương pháp giúp máy tính có khả năng "nhìn thấy", hiểu và diễn giải thông tin từ hình ảnh hoặc video giống như con người. Mục tiêu chính của thị giác máy tính là tự động hóa các tác vụ liên quan đến thị giác, chẳng hạn như nhận diện khuôn mặt, phát hiện vật thể, phân đoạn ảnh, theo dõi chuyển động, hoặc phân tích hành vi[4].

Trong các hệ thống giám sát thông minh, thị giác máy tính thường được kết hợp với camera và các thuật toán xử lý ảnh để thu thập, phân tích và nhận diện các đối tượng, sự kiện trong không gian giám sát. Ví dụ, một hệ thống bãi giữ xe thông minh có thể sử dụng camera để nhận diện biển số xe, xác định khu vực xe đỗ, và ghi lại thời điểm xe vào hoặc ra.

Camera giám sát là thiết bị phần cứng đóng vai trò thu/nhận hình ảnh hoặc video tại thời gian thực từ môi trường xung quanh. Dữ liệu thu được từ camera là nguồn đầu vào chính cho các thuật toán xử lý ảnh trong hệ thống thị giác máy tính. Trong các hệ thống thị giác máy tính, camera đóng vai trò như cảm biến chính, cung cấp đầu vào trực quan cho các thuật toán xử lý ảnh và mô hình học máy. Các camera này thường được tích hợp trong hệ thống an ninh, giao thông, giám sát công nghiệp và bãi đỗ xe thông minh. Các camera thường được sử dụng bao gồm:

- **Camera IP:** Truyền dữ liệu qua mạng, phù hợp với hệ thống giám sát quy mô lớn.
- **Camera USB:** Kết nối trực tiếp với máy tính hoặc vi điều khiển, thích hợp cho ứng dụng nhỏ và giá rẻ.

- **Camera tích hợp trên các bo mạch nhúng:** Như Raspberry Pi Camera, ESP32-CAM, Jetson Nano thường được sử dụng trong các dự án DIY và IoT.

Trong hệ thống giám sát bãi đỗ xe thông minh, camera thường được đặt ở vị trí có thể quan sát rõ biển số và vị trí xe, hỗ trợ việc nhận diện và lưu trữ thông tin liên quan đến phương tiện.

Xử lý ảnh cơ bản (Image Processing) là bước tiền xử lý quan trọng trong thị giác máy tính giúp cải thiện chất lượng hình ảnh, loại bỏ nhiễu, làm nổi bật đặc trưng và trích xuất thông tin cần thiết trước khi đưa vào mô hình nhận diện. Các bước xử lý ảnh cơ bản bao gồm:

- **Chuyển đổi ảnh màu sang ảnh xám (Grayscale conversion):** Từ ảnh RGB sang ảnh xám (grayscale) để giảm chiều dữ liệu và đơn giản hóa tính toán.
- **Lọc nhiễu (Noise Reduction):** Sử dụng các bộ lọc như Gaussian, Median để làm mịn ảnh và giảm nhiễu.
- **Nguộing hóa (Thresholding):** Phân biệt đối tượng và nền dựa trên giá trị cường độ điểm ảnh, có thể thực hiện bằng cách biến ảnh xám thành ảnh nhị phân.
- **Phát hiện biên (Edge Detection):** Dùng các thuật toán như Sobel, Canny để xác định ranh giới đối tượng trong ảnh.
- **Phát hiện và trích xuất đặc trưng (Feature Detection):** Sau khi tiền xử lý, ảnh sẽ được đưa vào các mô hình học máy hoặc mạng nơ-ron để phát hiện hoặc nhận dạng các điểm, đường, hình dạng đặc biệt để phục vụ nhận diện hoặc theo dõi...

Sự kết hợp giữa camera giám sát và xử lý ảnh cho phép các hệ thống thông minh như bãi đỗ xe có thể nhận diện biển số xe, phát hiện phương tiện, đếm số lượng xe ra vào và hỗ trợ giám sát an ninh một cách tự động và hiệu quả. Những kỹ thuật xử lý ảnh cơ bản này là nền tảng cho các bước phân tích cao hơn như phát hiện đối tượng, theo dõi chuyển động, hoặc phân tích hành vi.

2.5 PHÂN LOẠI ẢNH VỚI SUPPORT VECTOR CLASSIFIER (SVC)

Support Vector Classifier (SVC) là một biến thể của thuật toán Support Vector Machine (SVM), thường được sử dụng trong các bài toán phân loại. Trong hệ thống bãi đỗ xe thông minh, SVC đóng vai trò quan trọng trong việc phân loại hình ảnh khu vực đỗ xe để xác định xem vị trí đó đang trống hay đã có xe[5].

Nguyên lý hoạt động

SVC hoạt động bằng cách tìm một siêu phẳng (*hyperplane*) tối ưu trong không gian đặc trưng để phân tách các lớp dữ liệu với khoảng cách lớn nhất. Trong bài toán phân loại ảnh, các đặc trưng (*features*) đầu vào có thể được trích xuất từ ảnh đầu vào thông qua các kỹ thuật xử lý ảnh cơ bản như chuyển ảnh sang thang xám, làm mờ, phát hiện biên hoặc sử dụng histogram pixel[5].

Khi áp dụng cho ảnh camera của bãi giữ xe, mỗi ảnh sẽ được xử lý để trích xuất ra vector đặc trưng, từ đó SVC sử dụng các vector này để học và phân loại thành hai lớp: “Empty” và “Not Empty”.

Ưu điểm của SVC trong bài toán phân loại ảnh

- **Hiệu quả với dữ liệu có số chiều cao:** SVC hoạt động tốt trong các không gian đặc trưng có nhiều chiều, phù hợp với dữ liệu ảnh.
- **Khả năng tổng quát hóa tốt:** SVC thường cho độ chính xác cao với dữ liệu chưa thấy qua nhờ khoảng cách phân tách lớn.
- **Tối ưu khi dữ liệu không tuyến tính:** Sử dụng *kernel trick* (như RBF kernel) giúp SVC phân loại hiệu quả trong các bài toán không tuyến tính.

Ứng dụng trong đề tài

Trong đề tài này, SVC được huấn luyện trên tập dữ liệu hình ảnh từ camera (hoặc video demo thay thế) ghi lại các khu vực đỗ xe, sau đó được sử dụng để dự đoán trạng thái của từng ô đỗ. Kết quả phân loại sẽ là một trong hai lớp:

- **Label 0:** Vị trí trống (empty)
- **Label 1:** Vị trí có xe (not empty)

Kết quả đầu ra của mô hình sẽ được sử dụng để hiển thị tình trạng đỗ xe trên giao diện Dashboard và lưu trữ vào hệ thống cơ sở dữ liệu Firebase.

2.6 CƠ SỞ XỬ LÝ ẢNH BẰNG MASK VÀ PHÂN VÙNG ĐỊNH TRƯỚC TRONG PHÂN TÍCH BÃI ĐỖ XE

Trong hệ thống bãi đỗ xe thông minh được xây dựng, thay vì sử dụng các kỹ thuật xử lý ảnh phức tạp để phát hiện phương tiện, tác giả đã áp dụng một phương pháp đơn giản và hiệu quả hơn: sử dụng ảnh mask để đánh dấu sẵn các khu vực đỗ xe trong khung hình. Nhờ đó, hệ thống có thể xác định trạng thái các vị trí đỗ xe (có xe hoặc trống) một cách nhanh chóng và chính xác, mà không cần đến mô hình học sâu hay xử lý ảnh nâng cao.

Quy trình thực hiện gồm các bước chính:

- **Tạo mask đánh dấu vị trí đỗ xe:** Ảnh mask là ảnh đen trắng, trong đó mỗi khu vực đỗ xe được tô trắng (giá trị pixel khác 0). Từ mask này, sử dụng hàm `cv2.connectedComponentsWithStats`, hệ thống tự động trích xuất ra danh sách các bounding boxes tương ứng với từng vị trí đỗ xe.
- **Đọc video và trích xuất frame:** Hệ thống sử dụng một video mô phỏng cố định và lặp lại mỗi khi kết thúc video (dùng để mô phỏng cho video thực tế thu được bằng camera) để đọc từng khung hình. Mỗi vài chục khung hình, hệ thống thực hiện kiểm tra lại trạng thái từng vị trí.
- **Tính toán sự thay đổi (difference):** Để tiết kiệm tài nguyên, hệ thống so sánh sự thay đổi trung bình độ sáng giữa frame hiện tại và frame trước đó tại mỗi vị trí đỗ xe. Những vùng có thay đổi đủ lớn sẽ được kiểm tra kỹ hơn.
- **Phân loại chỗ trống:** Với các vùng nghi ngờ có sự thay đổi, hệ thống sử dụng hàm `empty_or_not` (được xây dựng riêng) để xác định xem tại vùng đó có xe hay không. Hàm này dựa trên đặc trưng cơ bản như độ sáng trung bình hoặc độ tương phản để đánh giá.
- **Cập nhật kết quả lên Firebase:** Số lượng vị trí trống và tổng số vị trí được cập nhật theo thời gian thực lên Firebase Realtime Database, phục vụ cho Dashboard giám sát.
- **Hiển thị trực quan lên video:** Hệ thống vẽ khung xanh cho vị trí trống và khung đỏ cho vị trí đã có xe ngay trên video, đồng thời dữ liệu số lượng ô trống và tổng số ô đỗ sẽ được cập nhật theo thời gian thực lên Firebase và sẽ được hiển thị trên góc màn hình Dashboard.

Phương pháp sử dụng mask có ưu điểm là **đơn giản, ít tốn tài nguyên và dễ triển khai**, phù hợp với hệ thống bãi đỗ xe cố định. Tuy nhiên, phương pháp này chỉ áp dụng hiệu quả trong điều kiện các ô đỗ xe đã được xác định trước và camera giữ nguyên góc quay. Trong tương lai, nếu hệ thống cần linh hoạt hơn (nhiều bãi xe khác nhau, thay đổi camera), có thể tích hợp thêm mô hình học máy hoặc xử lý ảnh nâng cao.

2.7 TỔNG QUAN VỀ JDANGO VÀ XÂY DỰNG GIAO DIỆN DASHBOARD

Django là một framework mã nguồn mở mạnh mẽ được phát triển bằng ngôn ngữ lập trình Python, chuyên dùng để xây dựng các ứng dụng web theo mô hình MVT (Model - View - Template). Django cung cấp một bộ công cụ toàn diện giúp lập trình viên phát triển nhanh chóng các hệ thống web có cấu trúc rõ ràng, dễ bảo trì và dễ mở rộng. Với các tính năng như hệ thống quản lý người dùng, routing, ORM (Object-Relational Mapping), bảo mật tích hợp và hỗ trợ quản trị web, Django được sử dụng rộng rãi trong các hệ thống yêu cầu truy xuất và xử lý dữ liệu thời gian thực[6].

Trong đề tài này, Django được sử dụng để xây dựng giao diện Dashboard quản lý bãi đỗ xe thông minh. Giao diện này có chức năng hiển thị các thông tin như:

- Danh sách xe ra vào cùng thời gian tương ứng.
- Hình ảnh được gửi từ hệ thống camera (hoặc mô phỏng ảnh/video).
- Tình trạng hiện tại của các ô đỗ xe (trống hoặc có xe).
- Thông tin người dùng/biển số xe nếu tích hợp chức năng nhận diện bổ sung.

Hệ thống Dashboard được triển khai với các thành phần chính như sau:

- **Frontend:** Sử dụng Django Template hoặc tích hợp với HTML/CSS/JS để hiển thị bảng điều khiển và tương tác người dùng.
- **Backend:** Django xử lý logic ứng dụng, nhận và xử lý dữ liệu từ các thiết bị phần cứng hoặc mô hình học máy.
- **Database/Firebase:** Lưu trữ thông tin UID, thời gian, trạng thái bãi xe và ảnh tại thời điểm xe ra/vào.

Việc sử dụng Django trong đề tài giúp đảm bảo tính linh hoạt, bảo mật và khả năng tích hợp dễ dàng với các hệ thống phần cứng thông qua giao tiếp API hoặc thư viện trung gian như `firebase-admin` (trong Python). Đồng thời, nó cũng hỗ trợ triển khai giao diện trực quan và thuận tiện cho việc giám sát, quản lý và mở rộng hệ thống trong tương lai.

Chương 3

THIẾT KẾ HỆ THỐNG

3.1 YÊU CẦU CỦA HỆ THỐNG

Hệ thống bãi đỗ xe thông minh cần đáp ứng được các yêu cầu kỹ thuật và chức năng nhằm đảm bảo hoạt động hiệu quả, chính xác và ổn định trong môi trường thực tế. Các yêu cầu này bao gồm:

Yêu cầu phần cứng

- **Vi điều khiển ESP8266 NodeMCU:** đóng vai trò trung tâm điều khiển và truyền dữ liệu giữa các cảm biến và hệ thống giám sát.
- **Module RFID RC522:** dùng để nhận diện phương tiện khi vào/ra bãi giữ xe thông qua thẻ RFID.
- **Camera giám sát** (phần này sẽ sử dụng video để mô phỏng camera giám sát): ghi nhận hình ảnh khu vực đỗ xe để phục vụ xử lý và phân tích bằng machine learning.
- **Màn hình hiển thị Dashboard:** cung cấp thông tin trực quan về trạng thái đỗ xe.
- **Nguồn điện ổn định:** đảm bảo hệ thống hoạt động liên tục.

Yêu cầu phần mềm

- **Mô hình machine learning (SVC):** được huấn luyện để phân loại trạng thái từng vị trí đỗ xe (trống hoặc có xe).
- **Hệ thống xử lý ảnh:** xử lý khung hình đầu vào để xác định vùng đỗ và trích xuất đặc trưng ảnh.

- **Firebase Realtime Database:** lưu trữ và đồng bộ dữ liệu thời gian thực như UID thẻ, biển số, trạng thái vị trí đỗ xe.
- **Giao diện Dashboard (Django):** sử dụng Django để tạo Dashboard hiển thị trạng thái các vị trí đỗ, lịch sử vào/ra xe, thông tin người dùng một cách trực quan.

Yêu cầu chức năng

- Hệ thống có khả năng nhận dạng xe ra vào thông qua RFID và lưu trữ UID, thời gian.
- Có thẻ phân tích ảnh/video để xác định chính xác các vị trí đang trống hoặc đã có xe dựa trên video quay được từ camera.
- Giao diện dashboard trực quan, dễ sử dụng cho người quản lý bãi xe.
- Đồng bộ dữ liệu thời gian thực giữa các thiết bị và hệ thống lưu trữ.
- Đảm bảo độ chính xác cao trong việc nhận diện và phân loại vị trí đỗ xe.

Yêu cầu phi chức năng

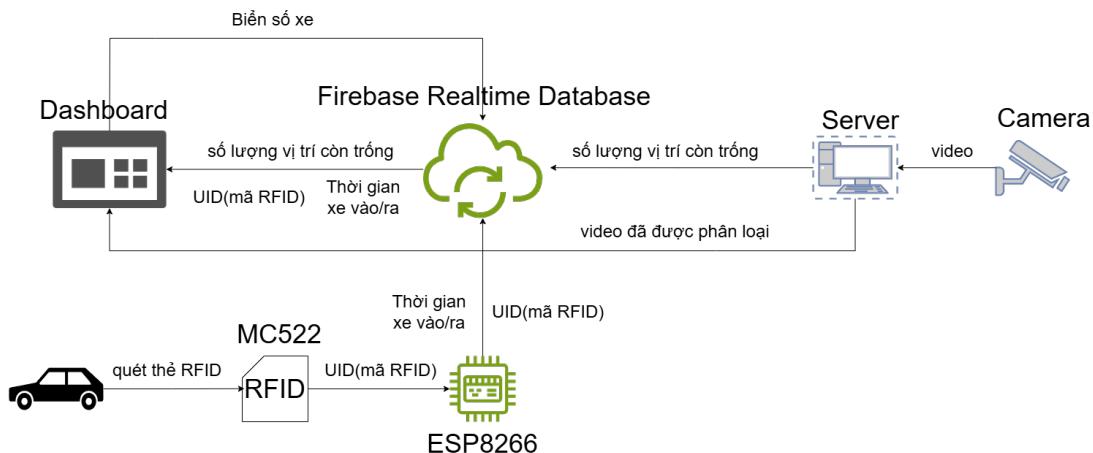
- Hệ thống phải có độ trễ thấp, đảm bảo phản hồi gần như tức thời.
- Dễ dàng mở rộng để áp dụng cho bãi xe có quy mô lớn hơn.
- Tiết kiệm năng lượng, hoạt động ổn định với các thiết bị công suất thấp.
- Chi phí hợp lý, phù hợp triển khai trong môi trường thực tế như trường học, bãi xe văn phòng, chung cư.

3.2 TỔNG QUAN KIẾN TRÚC HỆ THỐNG

Hệ thống bãi đỗ xe thông minh được xây dựng dựa trên sự kết hợp giữa phần cứng (ESP8266, module RFID RC522), phần mềm (giao diện Dashboard, Firebase) và công nghệ xử lý hình ảnh (camera giám sát và mô hình phân loại video). Mô hình hoạt động tổng quát được trình bày trong Hình 3.2.1.

Quy trình hoạt động gồm các bước chính như sau:

1. **Xe vào/ra bãi gửi – Quét thẻ RFID:** Khi một phương tiện vào bãi giữ xe, người dùng thực hiện quét thẻ RFID tại thiết bị đọc RC522. UID (mã định danh duy nhất của thẻ RFID) được đọc và truyền đến vi điều khiển ESP8266, kèm theo thời gian vào hoặc ra của xe.



Hình 3.2.1: Mô hình hoạt động của hệ thống

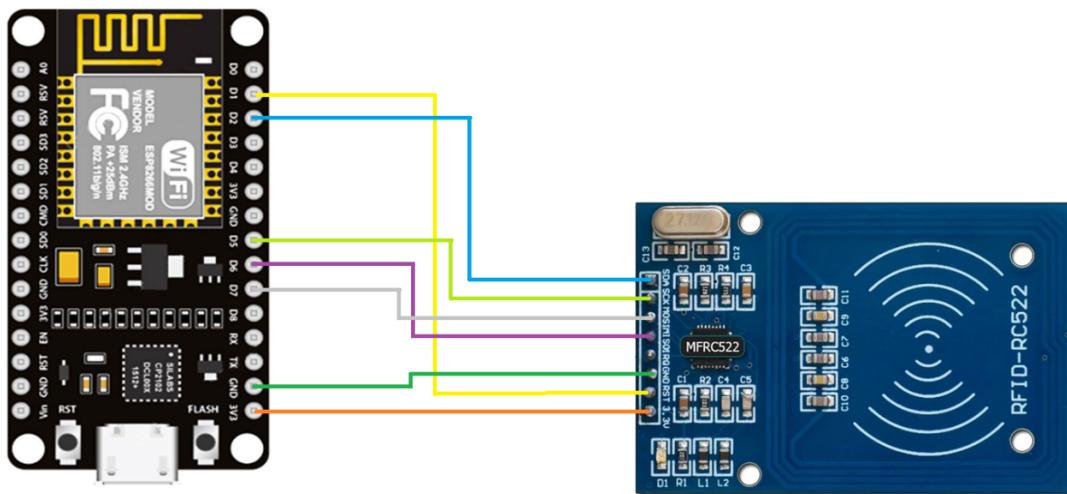
2. **Gửi dữ liệu lên Firebase:** ESP8266 kết nối WiFi và gửi dữ liệu gồm UID và thời gian vào/ra lên Firebase Realtime Database thông qua WiFi để lưu trữ và đồng bộ.
3. **Giám sát và phân tích bãi đỗ – Camera và Server:** Camera giám sát liên tục ghi lại video thời gian thực tại khu vực bãi đỗ xe và truyền tín hiệu đến Server để xử lý. Tại đây, hệ thống xử lý ảnh sẽ phân tích video để xác định số lượng vị trí còn trống, đồng thời phân loại và trích xuất thông tin liên quan. Kết quả sau xử lý được cập nhật lên Firebase Realtime Database để đồng bộ với các thành phần khác của hệ thống. Bên cạnh đó, video sau khi được xử lý cũng sẽ được truyền đến Dashboard và hiển thị trong mục *Parking Map*, giúp người quản lý giám sát bãi đỗ xe một cách trực quan và hiệu quả hơn.
4. **Dashboard hiển thị dữ liệu:** Giao diện Dashboard (xây dựng bằng Django) sẽ lấy dữ liệu từ Firebase và video từ server để hiển thị thông tin, bao gồm: UID và thời gian xe vào/ra, số lượng vị trí còn trống/tổng số lượng vị trí trong bãi đỗ xe, và thông tin biển số xe được nhập thủ công hoặc tích hợp nhận diện (sau khi được cài tiến).
5. **Tổng thể hệ thống:** Tất cả các thành phần đều được đồng bộ thông qua Firebase, giúp hệ thống cập nhật dữ liệu thời gian thực, giúp đảm bảo tính real-time trong việc cập nhật dữ liệu ra/vào và tình trạng bãi đỗ. Ngoài ra, mô hình còn có thể mở rộng để tích hợp thêm nhiều tính năng khác như nhận diện biển số tự động hoặc cảnh báo đỗ sai quy định.

3.3 THIẾT KẾ PHẦN CỨNG: ESP8266, MODULE RFID RC522 VÀ TRUYỀN DỮ LIỆU

Trong hệ thống bãi đỗ xe thông minh, phần cứng đóng vai trò quan trọng trong việc nhận diện phương tiện và truyền dữ liệu lên hệ thống. Hệ thống này được thiết kế nhằm thực hiện chức năng quét thẻ RFID tại các điểm vào/ra bãi đỗ xe, sau đó gửi thông tin lên cơ sở dữ liệu để xử lý và quản lý. Thành

phần chính trong hệ thống bao gồm:

- **Vi điều khiển ESP8266 NodeMCU:** Đóng vai trò là trung tâm xử lý tín hiệu và truyền dữ liệu từ thiết bị đọc thẻ RFID đến Firebase Realtime Database thông qua kết nối Wi-Fi. ESP8266 cũng được cấu hình để lấy thời gian thực và gửi kèm UID của thẻ vào cơ sở dữ liệu nhằm lưu trữ và theo dõi thời điểm xe vào/ra.
- **Thẻ RFID (MIFARE):** Mỗi thẻ có UID duy nhất, đại diện cho từng phương tiện
- **Thiết bị đọc thẻ Module RFID RC522:** Được sử dụng để đọc UID từ thẻ RFID. Thiết bị này giao tiếp với ESP8266 thông qua giao thức SPI. Sơ đồ đấu nối các chân được trình bày trong Hình 3.3.2.



Hình 3.3.2: Sơ đồ kết nối ESP8266 và Module RFID RC522

RC522 Pin	ESP8266 Pin
SDA	D2 (GPIO4)
SCK	D5 (GPIO14)
MOSI	D7 (GPIO13)
MISO	D6 (GPIO12)
IRQ	Not used
GND	GND
RST	D1 (GPIO5)
3.3V	3.3V

Bảng 3.3.1: Sơ đồ kết nối giữa RC522 và ESP8266

Nguyên lý hoạt động

Khi người dùng quét thẻ RFID, module RC522 sẽ đọc mã **UID** từ thẻ. Quá trình hoạt động của hệ thống được chia thành hai giai đoạn chính: *check-in* và *check-out*.

1. Check-in:

- Người dùng quét thẻ RFID tại điểm vào.
- Module RC522 đọc mã UID từ thẻ và gửi đến vi điều khiển ESP8266.
- ESP8266 nhận mã UID, gắn kèm thời gian thực (*timestamp*) và gửi dữ liệu này lên **Firebase Realtime Database**.
- Firebase lưu trữ dữ liệu và đồng bộ với các phần còn lại của hệ thống như Dashboard và server xử lý video.

2. Check-out:

- Khi người dùng quét lại cùng một thẻ RFID tại điểm ra, hệ thống xác định đó là lần quét thứ hai, tức là hành động *check-out*.
- ESP8266 gửi mã UID cùng thời gian check-out lên Firebase.
- Hệ thống tra cứu thời gian check-in tương ứng trong cơ sở dữ liệu, tính toán tổng thời gian gửi xe.
- Dựa vào thời gian sử dụng, hệ thống tính toán tổng số tiền cần thanh toán cho mã RFID đó.
- Nếu biển số xe đã được nhập qua Dashboard trước đó, hệ thống sẽ liên kết thông tin này với UID.
- Các thông tin gồm thời gian check-out, tổng thời gian gửi xe, tổng số tiền và biển số xe sẽ được cập nhật lên Dashboard để người quản lý hiển thị và xử lý.

Dữ liệu được lưu trữ trên Firebase bao gồm:

- UID của thẻ RFID.
- Thời gian xe vào/ra.
- Trạng thái bãi đỗ xe (có thể cập nhật số lượng chỗ trống dựa vào sự kiện vào/ra).
- Ngoài ra, hệ thống sử dụng node `latest_rfid` trong Firebase để lưu giá trị UID của thẻ vừa quét gần nhất. Node này hỗ trợ dashboard trong việc nhập thông tin biển số tương ứng cho xe mới check-in. Sau khi biển số được nhập xong, node `latest_rfid` có thể được cập nhật lại nhằm chuẩn bị cho lần quét tiếp theo.

Ưu điểm của thiết kế

Thiết kế hệ thống phần cứng và phần mềm như trên mang lại nhiều lợi ích:

- Hệ thống đơn giản, chi phí thấp và dễ mở rộng trong tương lai.
- ESP8266 tích hợp Wi-Fi, giúp loại bỏ nhu cầu sử dụng module mạng bổ sung.

- Module RC522 có tốc độ đọc nhanh, ổn định, và phù hợp cho các ứng dụng thực tế.
- Việc sử dụng Firebase Realtime Database cho phép hệ thống đồng bộ dữ liệu theo thời gian thực, giúp tiết kiệm chi phí triển khai server cục bộ.

Tóm lại, việc thiết kế và kết nối phần cứng như trên đảm bảo tính ổn định, chính xác trong quá trình nhận dạng phương tiện ra vào, đồng thời hỗ trợ truyền tải dữ liệu nhanh chóng đến hệ thống phần mềm để xử lý và hiển thị thông tin.

3.4 THIẾT KẾ PHẦN MỀM: GIAO DIỆN DASHBOARD SỬ DỤNG DJANGO, LƯU TRỮ VÀ QUẢN LÍ DỮ LIỆU BẰNG FIREBASE

Hệ thống phần mềm trong đề tài bao gồm ba thành phần chính: (1) giao diện Dashboard được xây dựng bằng framework *Django*, (2) cơ sở dữ liệu được xây dựng dựa trên *Firebase Realtime Database* để lưu trữ dữ liệu theo thời gian thực, và (3) mô hình học máy được tích hợp để phân tích hình ảnh từ camera được đặt trong bãi đỗ xe nhằm nhận diện tình trạng chỗ trống.

Giao diện Dashboard với Django

Giao diện Dashboard đóng vai trò là trung tâm giám sát và tương tác của hệ thống. Các chức năng chính bao gồm:

- Hiển thị thời gian hiện tại:** Dashboard hiển thị đồng hồ thời gian hiện tại được cập nhật liên tục, giúp người quản lý theo dõi chính xác thời điểm các sự kiện (check-in, check-out) diễn ra.
- Hiển thị thông tin xe gửi:** Danh sách các xe đang gửi trong dashboard tự động cập nhật liên tục thông qua dữ liệu từ Firebase. Mỗi dòng thông tin bao gồm UID thẻ RFID, biển số xe, thời gian check-in, check-out (nếu có) và tổng tiền gửi xe đã tính toán dựa trên thời gian gửi.
- Nhập biển số xe:** Sau khi người dùng quét thẻ RFID, hệ thống sẽ lưu lại UID tại node `latest_rfid` và chờ nhập biển số xe qua form trên Dashboard để lưu trữ cùng UID tương ứng và lưu trữ trong hệ thống.
- Hiển thị bản đồ trạng thái bãi đỗ (Parking Map):** Giao diện hiển thị sơ đồ các ô đỗ xe trực quan với trạng thái được mã hóa màu (xanh: trống, đỏ: có xe), các trạng thái này được cập nhật từ kết quả phân tích hình ảnh đầu vào của mô hình học máy.
- Hiển thị camera khuôn mặt và camera biển số xe của khách hàng khi check-in hoặc check-out:** Khi khách hàng quét thẻ tại điểm vào hoặc ra, hệ thống sẽ hiển thị hình ảnh khuôn

mặt và biển số xe (được chụp bởi camera hoặc trích xuất từ video mô phỏng) nhằm tăng tính xác thực và hỗ trợ công tác kiểm tra, đối chiếu.

Dashboard được phát triển bằng Django do ưu điểm dễ phát triển backend, hỗ trợ template mạnh và dễ tích hợp các API, JavaScript cũng như Firebase.

Lưu trữ và đồng bộ dữ liệu bằng Firebase Realtime Database

Firebase đóng vai trò là cơ sở dữ liệu trung gian kết nối giữa phần cứng (ESP8266, RFID RC522) và giao diện Dashboard. Dữ liệu được lưu tại các node chính:

- `latest_rfid`: lưu UID của thẻ RFID vừa quét để gán với biển số xe.
- `vehicles`: lưu thông tin chi tiết về các lượt gửi xe (UID, biển số, thời gian vào/ra, chi phí).
- `parking_lot_status`: lưu số ô đỗ còn trống trong bãi đỗ xe và tổng số lượng ô đỗ trong bãi, được cập nhật từ mô hình học máy.

Firebase Realtime Database cho phép đồng bộ dữ liệu theo thời gian thực, giảm độ trễ khi cập nhật trạng thái và đảm bảo giao diện người dùng luôn hiển thị dữ liệu mới nhất.

Tích hợp mô hình học máy nhận diện trạng thái bãi đỗ

Mô hình SVC được sử dụng để phân loại trạng thái từng vùng trong ảnh đầu vào thành “Empty” hoặc “Not Empty”. Quá trình tích hợp bao gồm:

1. Trích xuất từng vùng đã định nghĩa trước trong ảnh hoặc video mô phỏng dựa trên mask phân vùng sẵn từ trước.
2. Dự đoán trạng thái của từng vùng sử dụng mô hình đã huấn luyện.
3. Gửi kết quả lên Firebase theo định dạng phù hợp để Dashboard cập nhật. Kết quả được gửi chính là số ô đỗ còn trống trong bãi đỗ xe và tổng số lượng ô đỗ trong bãi

Giao tiếp và xử lý đồng bộ

Giao diện Dashboard sử dụng JavaScript để kết nối Firebase thông qua WebSocket và sự kiện `onValue()`, từ đó tự động cập nhật các thay đổi như:

- Có thẻ RFID mới được quét.

- Biển số xe được nhập.
- Trạng thái các vị trí đỗ xe thay đổi.

Việc cập nhật theo thời gian thực giúp nâng cao trải nghiệm người dùng và đảm bảo hệ thống hoạt động ổn định.

Minh họa giao diện

Hình ảnh minh họa Dashboard và sơ đồ Parking Map được trình bày lần lượt ở hình 4.3.2 và 4.2.1 bên dưới cho thấy sự tích hợp chặt chẽ giữa giao diện người dùng, cơ sở dữ liệu và mô hình nhận diện trạng thái đỗ xe.

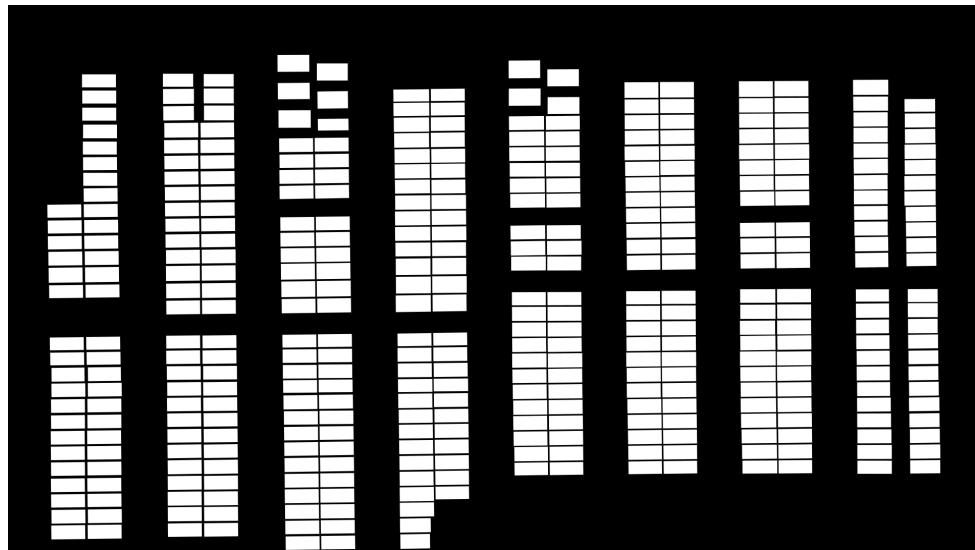
3.5 HỆ THỐNG PHÂN TÍCH TRẠNG THÁI BÃI ĐỖ XE

3.5.1 Phát hiện tình trạng chỗ trống bằng mặt nạ định trước (mask)

Để xác định trạng thái từng vị trí đỗ xe (trống hoặc đã có xe), hệ thống sử dụng phương pháp phân tích hình ảnh đầu vào thông qua mặt nạ định trước (*pre-defined mask*). Phương pháp này hoạt động dựa trên cơ sở các phân vùng đã được xác định sẵn tương ứng với từng ô đỗ xe trong bãi.

- **Chuẩn bị mặt nạ (mask):** Dựa trên hình ảnh toàn cảnh bãi đỗ xe (ảnh từ video hoặc ảnh tĩnh), hệ thống xây dựng một mặt nạ (mask) bao phủ tất cả các vị trí đỗ xe, mỗi vị trí là một vùng hình chữ nhật độc lập. Một ví dụ minh họa mặt nạ được sử dụng được trình bày ở Hình 3.5.3.
- **Trích xuất vùng ảnh tương ứng:** Với mỗi khung hình từ video giám sát hoặc ảnh tĩnh, hệ thống tiến hành trích xuất từng vùng ảnh tương ứng với các ô đỗ xe dựa vào mặt nạ.
- **Tiền xử lý ảnh:** Các vùng ảnh trích xuất được chuyển về kích thước chuẩn 15×15 pixel, chuyển sang ảnh xám (*grayscale*), và chuẩn hóa để làm đầu vào cho mô hình phân loại.
- **Phân loại trạng thái:** Mỗi vùng ảnh sau tiền xử lý được đưa vào mô hình học máy `model.py` đã được huấn luyện trước đó (trong đề tài sử dụng mô hình SVC – Support Vector Classifier) để xác định trạng thái: “Empty” (trống) (có xe) hoặc “Not Empty”.
- **Hiển thị kết quả:** Kết quả được hiển thị trực tiếp lên Dashboard dưới dạng Parking Map, trong đó mỗi ô đỗ được mã hóa màu (đỏ: có xe, xanh: trống) tương ứng với dự đoán của mô hình.

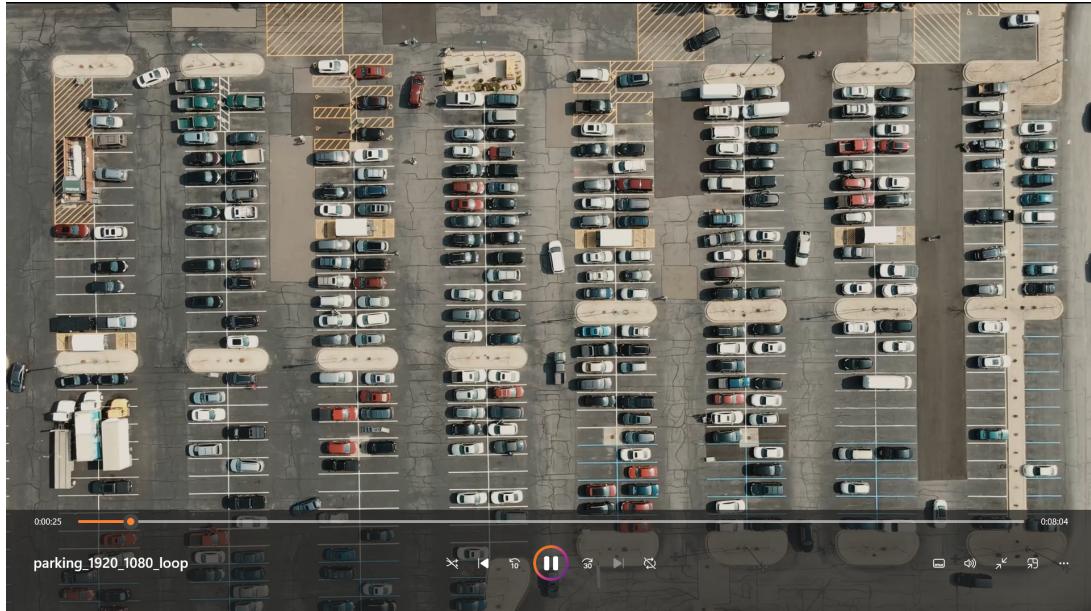
Phương pháp sử dụng mặt nạ định trước giúp đơn giản hóa quá trình phát hiện, giảm yêu cầu xử lý phức tạp so với các phương pháp nhận dạng toàn cảnh, đồng thời dễ dàng triển khai trên các hệ thống phần cứng có tài nguyên hạn chế.



Hình 3.5.3: Mặt nạ định trước (mask) bao gồm các phân vùng tương ứng với từng ô đỗ xe trong bãi

3.5.2 Mô phỏng video thay thế camera thực tế

Trong phạm vi đề tài, do điều kiện triển khai thực tế còn hạn chế, hệ thống không sử dụng camera giám sát thực tế để thu nhận dữ liệu hình ảnh từ bãi đỗ xe, mà thay vào đó là sử dụng các đoạn video mô phỏng để thay thế luồng hình ảnh đầu vào. Hình 3.5.4 dưới đây thể hiện một khung hình được trích xuất từ video mô phỏng sử dụng trong hệ thống.



Hình 3.5.4: Minh họa khung hình trích từ video mô phỏng bãi đỗ xe

Video mô phỏng được xây dựng dựa trên kịch bản thực tế tại bãi đỗ xe, bao gồm các tình huống có xe ra vào, thay đổi trạng thái ô đỗ, góc nhìn từ trên cao mô phỏng camera lắp đặt cố định. Mỗi khung hình (frame) trong video đều chứa đầy đủ thông tin về toàn cảnh bãi xe để hệ thống có thể xử lý và phát hiện trạng thái các ô đỗ.

Quy trình xử lý được thực hiện như sau:

- **Đọc video và tách khung hình:** Video đầu vào được đọc bằng thư viện xử lý ảnh OpenCV. Hệ thống thực hiện tách từng khung hình từ video ở tốc độ định sẵn (ví dụ: 1 frame/giây) để làm đầu vào cho mô hình phân tích.
- **Xử lý và phân tích mỗi frame:** Mỗi khung hình được cắt theo các vị trí định sẵn trên mặt nạ (mask) tương ứng với từng ô đỗ xe. Các vùng ảnh này sau đó được đưa vào mô hình phân loại trạng thái (được trình bày trong mục 3.5.1).
- **Cập nhật trạng thái bãi đỗ xe:** Dựa trên kết quả phân loại của mô hình (Empty/Not Empty), hệ thống cập nhật giao diện Parking Map và Dashboard, mô phỏng như đang làm việc với dữ liệu thời gian thực từ camera.
- **Hiển thị quá trình:** Để trực quan hóa quá trình hoạt động, mỗi frame được hiển thị cùng trạng thái từng ô đỗ. Màu sắc (xanh/dỏ) phản ánh kết quả phân loại từ mô hình. Video sau khi được xử lý sẽ được truyền đến Dashboard và hiển thị ở tab **Parking Map** như Hình 4.2.1

Cách tiếp cận này giúp tác giả có thể xây dựng, huấn luyện và kiểm tra mô hình học máy một cách linh hoạt mà không cần triển khai phần cứng camera thực tế trong giai đoạn đầu. Đồng thời, mô hình vẫn đảm bảo khả năng hoạt động ổn định và có thể dễ dàng chuyển sang môi trường thực tế trong các bước phát triển tiếp theo.

Chương 4

TRIỂN KHAI HỆ THỐNG THỰC TẾ

4.1 QUY TRÌNH TRIỂN KHAI HỆ THỐNG

Quy trình triển khai hệ thống “Bãi đỗ xe thông minh ứng dụng IoT và tích hợp machine learning nhận diện vị trí chỗ trống trong bãi đỗ xe” được thực hiện qua các bước sau đây:

- Xây dựng và cấu hình hệ thống phần cứng** Phần cứng của hệ thống bao gồm các thiết bị IoT như module ESP8266, RFID RC522 và các cảm biến hỗ trợ nhận diện tình trạng bãi đỗ xe. Đầu tiên, cần thiết lập các thiết bị phần cứng và kết nối chúng với nhau thông qua các giao thức truyền thông phù hợp (ví dụ: Wi-Fi cho ESP8266, I2C hoặc SPI cho các cảm biến). Module ESP8266 sẽ truyền tải dữ liệu từ các cảm biến RFID về cơ sở dữ liệu Firebase để theo dõi tình trạng xe trong bãi.
- Cài đặt và cấu hình Firebase Realtime Database** Firebase Realtime Database được sử dụng để lưu trữ và quản lý dữ liệu liên quan đến thông tin xe (biển số, UID RFID, thời gian vào/ra bãi) và trạng thái hiện tại của bãi đỗ xe (ví dụ như còn trống bao nhiêu chỗ/tổng số vị trí đỗ xe trong bãi). Cấu hình Firebase phải đảm bảo khả năng xử lý dữ liệu nhanh chóng và chính xác, đồng thời hỗ trợ việc cập nhật và truy xuất dữ liệu theo thời gian thực.
- Phát triển và triển khai mô hình nhận diện vị trí chỗ trống** Mô hình học máy dựa trên Support Vector Classifier (SVC) được huấn luyện với các ảnh mô phỏng bãi đỗ xe, phân biệt các khu vực có xe và các khu vực trống. Sau khi huấn luyện xong, mô hình được tích hợp vào hệ thống và triển khai để nhận diện các vị trí đỗ xe trống. Quá trình này cũng bao gồm việc xây dựng bộ dữ liệu huấn luyện và đánh giá độ chính xác của mô hình.
- Phát triển giao diện Dashboard sử dụng Django** Giao diện Dashboard được phát triển với

Django để người quản lý có thể dễ dàng theo dõi tình trạng bãi đỗ xe. Dashboard hiển thị các thông tin như biển số xe đã check-in/check-out, tình trạng từng ô đỗ xe (hiển thị qua Parking Map), thời gian hiện tại, Tất cả dữ liệu này được đồng bộ hóa với Firebase để cung cấp thông tin chính xác và thời gian thực.

5. **Kết nối và tích hợp hệ thống phần mềm và phần cứng** Sau khi các thành phần phần cứng và phần mềm được phát triển và kiểm tra độc lập, chúng được kết nối với nhau trong một môi trường thực tế. ESP8266 sẽ thu thập dữ liệu từ thẻ RFID, gửi dữ liệu này lên Firebase Realtime Database, và đồng thời kiểm tra trạng thái của từng ô đỗ xe thông qua mô hình nhận diện đã huấn luyện. Dữ liệu từ Firebase sẽ được cập nhật liên tục trên Dashboard để người quản lý có thể theo dõi và điều hành bãi đỗ xe một cách hiệu quả.
6. **Kiểm thử và tối ưu hệ thống** Sau khi kết nối hệ thống, việc kiểm thử sẽ được thực hiện để đảm bảo các chức năng của hệ thống hoạt động ổn định và chính xác. Các vấn đề phát sinh trong quá trình thử nghiệm như độ trễ trong việc cập nhật dữ liệu, chính xác của mô hình nhận diện vị trí đỗ xe, hoặc vấn đề kết nối sẽ được phân tích và tối ưu.

Quy trình triển khai này đảm bảo rằng hệ thống sẽ có thể hoạt động hiệu quả trong môi trường thực tế, đáp ứng nhu cầu theo dõi và quản lý bãi đỗ xe thông minh một cách tự động và chính xác.

4.2 HUẤN LUYỆN VÀ ĐÁNH GIÁ MÔ HÌNH HỌC MÁY

4.2.1 Mô tả dữ liệu đầu vào và quá trình huấn luyện SVC

Để nhận diện tình trạng chỗ đỗ (*trống* hoặc *đã có xe*), nhóm đã thu thập tập dữ liệu hình ảnh mô phỏng bãi đỗ xe, trong đó mỗi vùng đỗ được trích xuất và gán nhãn thủ công thành hai lớp:

- **empty:** vị trí không có xe
- **not_empty:** vị trí đã có xe

Các ảnh vùng đỗ được tiền xử lý theo kích thước chuẩn 15×15 pixel, sau đó được làm phẳng thành các vector đặc trưng đầu vào cho mô hình học máy. Toàn bộ dữ liệu hình ảnh được lưu trữ trong hai thư mục tương ứng với hai lớp `empty` và `not_empty`.

Dữ liệu sau xử lý được chia thành tập huấn luyện và kiểm tra với tỷ lệ 80/20 bằng hàm `train_test_split` từ thư viện `sklearn`, đảm bảo phân bố đồng đều giữa hai lớp nhờ tham số `stratify`.

Nhóm sử dụng mô hình **Support Vector Classifier (SVC)** – một biến thể của SVM phù hợp cho bài toán phân loại nhị phân với số chiều đặc trưng nhỏ. Quá trình huấn luyện bao gồm bước chọn

siêu tham số (hyperparameter tuning) bằng phương pháp GridSearchCV, thử nghiệm các giá trị của C và gamma để tìm mô hình tối ưu.

Dưới đây là đoạn code Python sử dụng để huấn luyện mô hình:

```

import numpy as np
import os
import pickle
from skimage.io import imread
from skimage.transform import resize
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report

data_dir = 'clf-data'
categories = ['empty', 'not_empty']

data = []
labels = []
for category_idx, category in enumerate(categories):
    for file in os.listdir(os.path.join(data_dir, category)):
        img_path = os.path.join(data_dir, category, file)
        img = imread(img_path)
        img = resize(img, (15, 15))
        data.append(img.flatten())
        labels.append(category_idx)

data = np.asarray(data)
labels = np.asarray(labels)

x_train, x_test, y_train, y_test = train_test_split(
    data, labels, test_size=0.2, shuffle=True, stratify=labels)

parameters = {
    'gamma': [0.01, 0.001, 0.0001],
    'C': [1, 10, 100, 1000],
}

model = GridSearchCV(SVC(), parameters, n_jobs=-1, verbose=1)
model.fit(x_train, y_train)

best = model.best_estimator_
y_predict = best.predict(x_test)
print(classification_report(y_test, y_predict))
pickle.dump(best, open('../model.p', 'wb'))

```

Sau khi huấn luyện, mô hình đạt kết quả phân loại tốt trên tập kiểm tra, thể hiện qua các chỉ số như độ chính xác (accuracy), độ nhạy (recall), và độ chính xác dự đoán (precision). Mô hình sau đó được

lưu lại dưới dạng file `model.p` để sử dụng trong bước triển khai nhận diện chỗ đỗ xe ở phần tiếp theo.

4.2.2 Triển khai mô hình nhận diện vị trí đỗ xe

Sau khi mô hình phân loại bằng SVC được huấn luyện và đánh giá ở bước trước, hệ thống được triển khai để nhận diện vị trí chỗ đỗ xe theo thời gian thực (hoặc từ video mô phỏng) từ model `model.py`. Mỗi khung hình từ camera được xử lý bằng cách cắt ra các vùng tương ứng với từng vị trí đỗ xe, dựa trên mặt nạ định trước (mask). Các vùng ảnh này sau đó được đưa vào mô hình để phân loại trạng thái là **trống** (Empty) hoặc **đã có xe** (Not Empty)[7].

Để thực hiện việc triển khai mô hình vào hệ thống, các bước xử lý ảnh được thực hiện như sau:

- Cắt từng vùng đỗ xe từ khung hình (sử dụng mask phân vùng định trước).
- Resize vùng ảnh về kích thước 15×15 pixels để phù hợp với kích thước dữ liệu đã huấn luyện.
- Làm phẳng ảnh về dạng vector 1 chiều để đưa vào mô hình.
- Dự đoán trạng thái bằng mô hình đã lưu (`model.p`) và cập nhật trạng thái hiển thị trên Dashboard.

Việc triển khai mô hình sử dụng thư viện `pickle` để load mô hình đã huấn luyện từ bước trước, cho phép hệ thống thực hiện dự đoán mà không cần huấn luyện lại mô hình từ đầu.

Đây là đoạn code minh họa quá trình dự đoán trạng thái chỗ đỗ xe:

Listing 4.1: Dự đoán trạng thái chỗ đỗ xe bằng mô hình đã huấn luyện

```
import pickle
from skimage.io import imread
from skimage.transform import resize
import numpy as np

model = pickle.load(open('./model.p', 'rb'))

img = imread('slot_example.jpg')
img = resize(img, (15, 15))
img_flat = img.flatten().reshape(1, -1)

prediction = model.predict(img_flat)

if prediction[0] == 0:
    print("Status: Empty")
else:
    print("Status: Not empty")
```

Sau khi dự đoán trạng thái của từng vùng ảnh, hệ thống sẽ cập nhật số lượng chỗ trống/tổng số lượng trong bãi gửi xe và gửi thông tin lên Firebase Realtime Database để đồng bộ với Dashboard. Người

quản lý có thể theo dõi tình trạng bãi xe trực tiếp trên giao diện web và xác định các vị trí còn trống để điều hướng xe vào bãi.

Nhờ tính chất nhẹ và hiệu quả của mô hình SVC với ảnh đầu vào kích thước nhỏ, hệ thống có thể hoạt động trên các thiết bị nhúng hoặc server cấu hình thấp mà vẫn đảm bảo tốc độ phản hồi tốt và độ chính xác cao.

4.2.3 Đánh giá độ chính xác mô hình nhận diện vị trí đỗ xe

Sau quá trình huấn luyện và triển khai, mô hình phân loại vị trí đỗ xe sử dụng kỹ thuật Machine Learning (SVC - Support Vector Classifier) đã cho thấy hiệu suất ổn định và độ chính xác cao trong việc phân biệt hai trạng thái **Empty** và **Not Empty** của các ô đỗ xe. Hệ thống có thể nhận diện chính xác vị trí có hoặc không có xe với độ trễ rất thấp trong việc xử lý và truyền dữ liệu.

Thời gian truyền dữ liệu từ thiết bị xử lý lên cơ sở dữ liệu Firebase Realtime Database là ngắn và gần như tức thì, giúp đảm bảo tính cập nhật liên tục của trạng thái bãi đỗ xe. Tuy nhiên, độ trễ vẫn có thể bị ảnh hưởng phần nào bởi tốc độ mạng Wi-Fi, do mô hình đang được triển khai và xử lý tại máy tính cục bộ (local server).

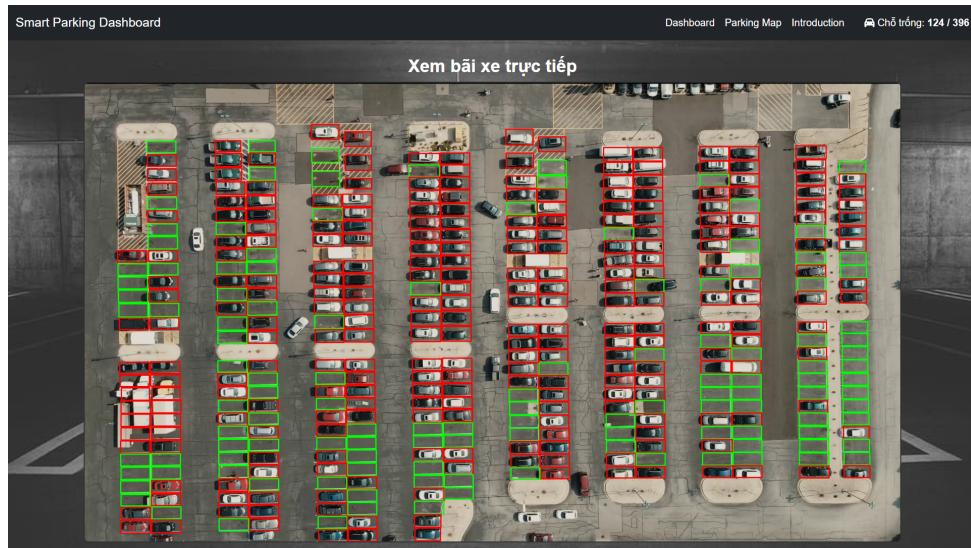
Video mô phỏng trạng thái bãi đỗ sau phân loại được hiển thị sắc nét, không gặp hiện tượng nhiễu, trễ hình hoặc bỏ sót (miss) các ô xe. Điều này cho thấy hệ thống xử lý ảnh, phân vùng mask và mô hình phân loại đã hoạt động đồng bộ và hiệu quả.

Nhờ việc sử dụng mô hình Machine Learning đơn giản, nhẹ và ít tài nguyên như SVC, hệ thống hoàn toàn có thể triển khai trên các thiết bị nhúng (như Raspberry Pi, Jetson Nano) hoặc các server cấu hình thấp. Điều này vừa giúp tiết kiệm chi phí phần cứng, vừa vẫn đảm bảo hiệu quả trong việc phản hồi nhanh và nhận diện chính xác trạng thái của bãi đỗ xe theo thời gian thực.

Hình 4.2.1 dưới đây minh họa giao diện bản đồ bãi đỗ xe sau khi hệ thống đã phân loại từng ô là **Empty** hoặc **Not Empty**, giúp người dùng có thể quan sát trực quan trạng thái chỗ đậu xe tại thời điểm thực.

4.3 TRIỂN KHAI KẾT NỐI THIẾT BỊ THỰC TẾ VÀ HỆ THỐNG DASHBOARD

Sau khi hoàn tất quá trình huấn luyện mô hình học máy và kiểm thử các thành phần của hệ thống, bước tiếp theo là tích hợp và kết nối toàn bộ thiết bị phần cứng với hệ thống phần mềm hiển thị – Dashboard như đã được trình bày trong sơ đồ 3.2.1 đã được trình bày ở phần 3.2 bên trên. Mục tiêu



Hình 4.2.1: Bản đồ vị trí chỗ đỗ xe (Parking Map) hiển thị trạng thái Empty và Not Empty

chính của phần này là đảm bảo rằng dữ liệu từ thiết bị RFID, trạng thái bãi đỗ (do mô hình máy học phân tích từ video mô phỏng) và giao diện Dashboard được đồng bộ, hoạt động theo thời gian thực và dễ dàng sử dụng.

Kết nối phần cứng với hệ thống qua Firebase

Các thiết bị gồm ESP8266 và module RFID RC522 được lập trình để đọc UID từ thẻ RFID khi người dùng đưa thẻ vào đầu đọc. Sau khi nhận diện, ESP8266 gửi dữ liệu UID và thời gian quét lên *Firebase Realtime Database* tại node `latest_rfid`. Tại thời điểm này, nếu người dùng nhập biển số xe thông qua Dashboard, dữ liệu sẽ được gán kèm với UID tương ứng, giúp quản lý được thông tin phương tiện.

Giao tiếp với Dashboard và hiển thị dữ liệu

Dashboard được xây dựng bằng Django và sử dụng JavaScript để lắng nghe các thay đổi từ Firebase theo thời gian thực. Khi có dữ liệu UID mới từ node `latest_rfid`, giao diện sẽ tự động cập nhật bảng thông tin xe gửi bao gồm:

- UID thẻ RFID
- Biển số xe (nếu đã được nhập)
- Thời gian check-in
- Thời gian check-out (có sau khi quét thẻ RFID lần 2)
- Tổng tiền phải trả (được tính khi check-out ra khỏi bãi)

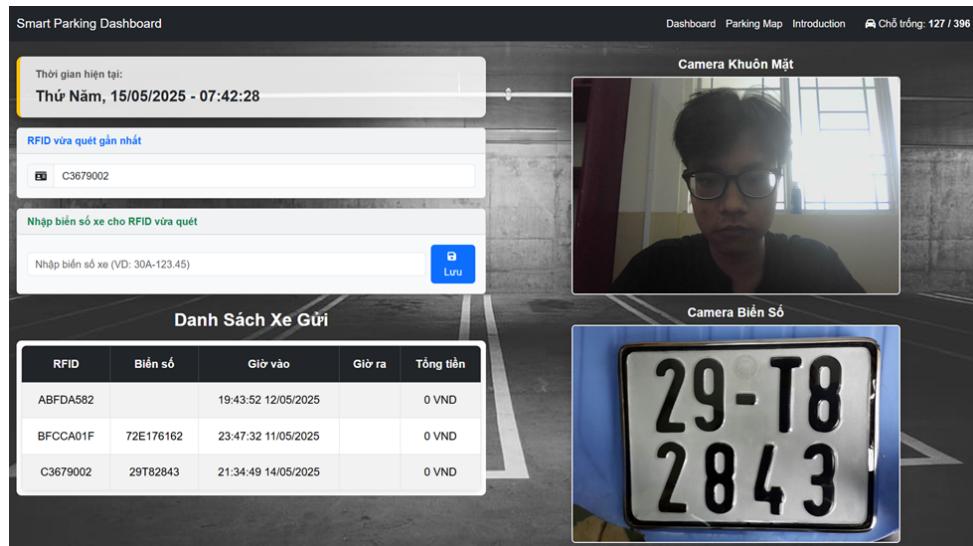
Hệ thống còn hỗ trợ theo dõi lịch sử gửi xe, giúp người dùng và quản trị viên truy xuất lại thông tin khi cần thiết.

Tích hợp với mô hình phân tích trạng thái bãi đỗ

Mô hình phân loại SVC được huấn luyện để nhận diện trạng thái từng vị trí trong bãi đỗ (trống hoặc có xe) từ video mô phỏng. Kết quả phân tích trạng thái được hiển thị trên Dashboard dưới dạng:

- Bản đồ trạng thái bãi đỗ (parking map):** Các ô đỗ xe được đánh dấu màu xanh (trống) hoặc đỏ (có xe) để dễ dàng quan sát.
- Danh sách trạng thái vị trí:** Mỗi vị trí được liệt kê kèm nhãn trạng thái, có thể tích hợp thêm thời gian cập nhật gần nhất.

Ảnh minh họa sơ đồ bãi đỗ đã được thể hiện ở 4.2.1 (đã được trình bày ở phần 4.2.3) cùng với giao diện Dashboard được mô tả ở 4.3.2 bên dưới, cho thấy sự đồng bộ giữa phần nhận diện video và giao diện người dùng.



Hình 4.3.2: Minh họa Dashboard bãi đỗ xe thông minh ứng dụng IoT và tích hợp machine learning nhận diện vị trí chỗ trống trong bãi đỗ xe

Hiệu suất và phản hồi thời gian thực

Hệ thống sử dụng Firebase Realtime Database làm cầu nối trung gian giữa phần cứng và Dashboard. Với Firebase, dữ liệu được cập nhật và đồng bộ hóa gần như tức thì, giúp giảm thiểu độ trễ trong toàn hệ thống. Qua kiểm thử thực tế, thời gian từ khi quét thẻ RFID đến khi thông tin hiển thị trên Dashboard thường nhỏ hơn 1 giây.

Chương 5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỀN

5.1 KẾT LUẬN

Đề tài “Bãi đỗ xe thông minh ứng dụng IoT và tích hợp Machine Learning nhận diện vị trí chỗ trống trong bãi đỗ xe” đã được triển khai và thử nghiệm với kết quả khả quan, đáp ứng được các mục tiêu ban đầu đề ra. Cụ thể:

- **Về phần cứng:** Hệ thống sử dụng vi điều khiển ESP8266 kết hợp với module RFID RC522 đã hoạt động ổn định, ghi nhận chính xác UID của thẻ và thời gian xe ra vào. Dữ liệu được truyền nhanh chóng và đồng bộ với Firebase Realtime Database, đảm bảo việc cập nhật thời gian thực.
- **Về phần mềm:** Giao diện Dashboard xây dựng bằng Django có khả năng hiển thị đầy đủ các thông tin quan trọng như: biển số xe, thời gian xe vào/ra, số lượng chỗ còn trống/tổng số vị trí đỗ, cùng với bản đồ bãi xe được hiển thị trong (*Parking Map*) trực quan. Điều này giúp người quản lý dễ dàng theo dõi và điều hành hoạt động bãi giữ xe.
- **Về phân tích video:** Hệ thống xử lý ảnh bằng kỹ thuật phân vùng mặt nạ định trước (mask) và mô hình phân loại SVC cho kết quả phân tích chính xác trên video mô phỏng, hỗ trợ hiệu quả trong việc phát hiện và phân loại tình trạng chỗ đỗ.

Nhìn chung, hệ thống đã chứng minh được tính ứng dụng thực tiễn, kết hợp hiệu quả giữa công nghệ IoT và học máy trong việc quản lý bãi xe thông minh, đồng thời có khả năng mở rộng và tích hợp thêm các chức năng mới trong tương lai.

5.2 ĐỀ XUẤT CẢI TIẾN VÀ MỞ RỘNG HỆ THỐNG TRONG TƯƠNG LAI

Mặc dù hệ thống đã đạt được những kết quả tích cực trong phạm vi thử nghiệm, vẫn còn nhiều hướng cải tiến và mở rộng để nâng cao tính hiệu quả, độ chính xác và khả năng triển khai thực tế trong các bối cảnh công cộng. Một số đề xuất cụ thể như sau:

- **Tích hợp camera thời gian thực:** Thay vì sử dụng video mô phỏng, hệ thống có thể được nâng cấp để sử dụng camera thực tế gắn trực tiếp tại bãi đỗ xe. Điều này giúp hệ thống hoạt động trong điều kiện thực tiễn, phù hợp hơn với các ứng dụng thực tế và cải thiện độ chính xác trong phân tích hình ảnh.
- **Tích hợp hệ thống nhận diện biển số xe (ALPR):** Hệ thống có thể bổ sung thêm module nhận diện biển số xe (Automatic License Plate Recognition) thông qua camera. Việc này cho phép tự động ghi nhận biển số xe khi vào/ra, hỗ trợ tra cứu, định danh phương tiện nhanh chóng và có thể kết hợp với hệ thống kiểm soát an ninh hoặc thanh toán tự động.
- **Áp dụng các mô hình học sâu (Deep Learning):** Mô hình SVC tuy có ưu điểm đơn giản và nhẹ, nhưng độ chính xác còn phụ thuộc nhiều vào kỹ thuật tiền xử lý. Trong tương lai, có thể thay thế bằng các mô hình học sâu như CNN (Convolutional Neural Networks) để cải thiện hiệu suất phân loại và nhận diện chố đỗ trong môi trường phức tạp.
- **Tối ưu giao diện Dashboard:** Dashboard có thể được cải tiến bằng cách bổ sung các biểu đồ thống kê, cảnh báo khi xe gửi quá thời gian, hoặc hỗ trợ tìm kiếm xe theo mã RFID hoặc biển số để phục vụ quản lý hiệu quả hơn.
- **Bổ sung xác thực an ninh:** Để tăng tính bảo mật, hệ thống có thể tích hợp thêm các cơ chế xác thực người dùng như mã OTP (One Time Password), nhận diện khuôn mặt hoặc tích hợp mã QR (Quick Response Code) song song với thẻ RFID.
- **Mở rộng quy mô hệ thống:** Trong giai đoạn tiếp theo, hệ thống có thể được triển khai trên quy mô lớn với nhiều điểm vào/ra và nhiều camera giám sát, phục vụ cho các bãi giữ xe lớn tại trung tâm thương mại, khu công nghiệp hoặc khuôn viên trường đại học.
- **Tối ưu tài nguyên phần cứng:** Tùy vào số lượng người dùng và thiết bị, hệ thống có thể chuyển sang sử dụng ESP32 với hiệu năng cao hơn, đồng thời áp dụng các phương pháp nén và truyền dữ liệu hiệu quả để giảm tải băng thông và tiết kiệm chi phí.

Những đề xuất trên nếu được triển khai đồng bộ sẽ giúp hệ thống không chỉ dừng lại ở mức thử nghiệm, mà có thể áp dụng trong thực tế với hiệu quả cao, góp phần giải quyết bài toán quản lý bãi đỗ xe trong đô thị thông minh.

TÀI LIỆU THAM KHẢO

- [1] Espressif Systems, “ESP8266EX Datasheet,” <https://www.espressif.com/en/products/socs/esp8266>, accessed: May. 2025.
- [2] M. A. Mikki, “Rfid technology and applications,” in *Proc. Int. Conf. RFID-Technologies and Applications (RFID-TA)*. IEEE, 2011, pp. 1–5.
- [3] Firebase, “Firebase Realtime Database,” <https://firebase.google.com/docs/database>, accessed: May. 2025.
- [4] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Springer, 2022.
- [5] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [6] Django Software Foundation, “Django documentation,” <https://docs.djangoproject.com/>, accessed: May. 2025.
- [7] Computer vision engineer, “Smart parking system using iot and machine learning,” <https://www.youtube.com/watch?v=F-884J2mnOY>, 2022, truy cập ngày 10/4/2025. [Online]. Available: <https://www.youtube.com/watch?v=F-884J2mnOY>